VOLUME 5. NO. 1.

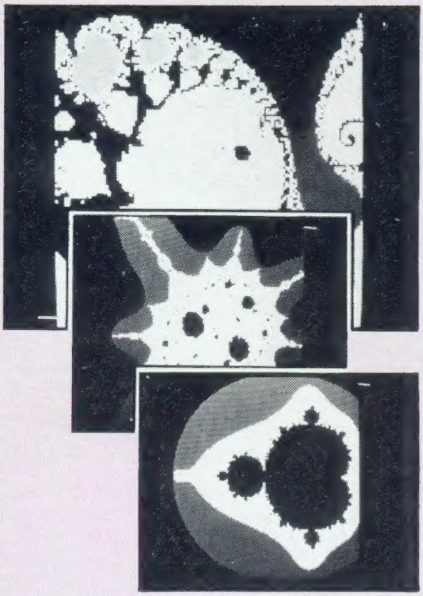
MAY 1986 PRICE £1.30

BEEBUG

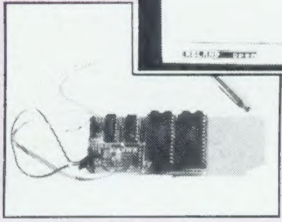
FOR THE BBC MICRO

MANDLEBROT GRAPHICS

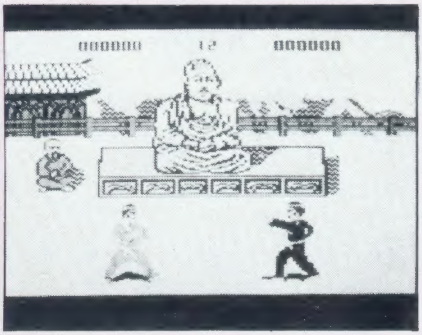
Mandelbrot



The Phone Book



Sideways RAM Modules



Martial Arts

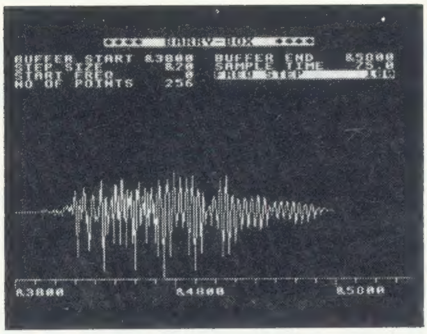
BEEBUG

VOLUME 5 NUMBER 1 MAY 198

GENERAL CONTENTS

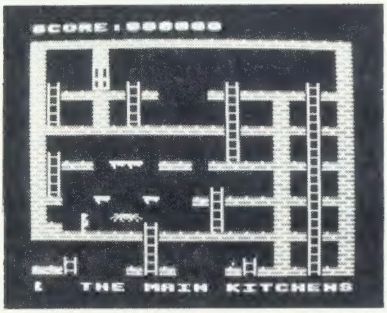
PROGRAMS

HINTS AND TIPS



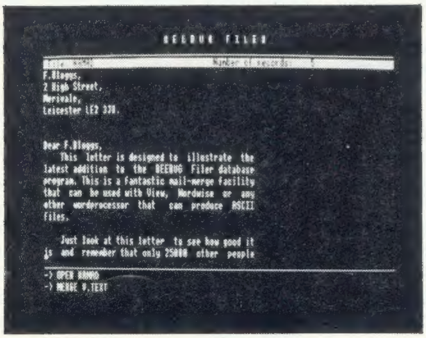
```
****  BARRY-BOX  ****
BUFFER START &3800   BUFFER END    &5800
STEP SIZE      &20   SAMPLE TIME    75.0
START FREQ       0   FREQ STEP       100
NO OF POINTS   256
```

The Barry Box



Ebony Castle

Printer Survey



```
BEEBUG FILER
File: NAME                 Number of records:  5
F.Bloggs,
2 High Street,
Merivale,
Leicester LE2 3JR.


Dear F.Bloggs,
   This letter is designed to illustrate the
latest addition to the BEEBUG Filer database
program. This is a fantastic mail-merge facility
that can be used with View, Wordwise or any
other wordprocessor that can produce ASCII
files.

   Just look at this letter to see how good it
is and remember that only 25000 other people
-> OPEN NAMES
-> MERGE V.TEXT
```

Mail-Merge with Filer

EDITORIAL JOTTINGS

SPECIAL OFFERS

Included with this, the first issue of volume 5 of BEEBUG, is a complete index to volume 4.

This has been carefully arranged to assist you in looking up the wealth of information that we have published over the last 12 months.

Of course, if you want a really comprehensive and fast access index to all the past issues of BEEBUG then Magscan, our magazine bibliography on a disc, will be the answer. The whole of volume 4 is now available on a single disc to add to the original volumes 1 to 3. This is available now from our High Wycombe address at a cost of £4.50 plus 50p post & packing.

One of the most fascinating devices to have come our way for some time is the Barry Box (designed and made by Barry would you believe it) which we review in this issue. We have arranged a special discount price for BEEBUG members of just £71.95, saving £8 on the normal retail price. The Barry Box is available direct from the supplier only (see review for address) and you must quote your membership number with your order. The special members price will be available only for a limited period so order now.

MAGAZINE CASSETTE/DISC

This month's magazine cassette/disc has to be one of the best yet for value with a total of 20 programs and files. Apart from all the other programs, it includes the complete BEEBUG Filer program with the latest update, a mail-merge option, and the USI utilities (if you missed them before) which are the basis of the Phone Book application described this month. All this for just £3.00 (cassette) or £4.75 (disc) plus 50p post and packing. Full details are on the inside back cover.

BEEBUGSOFT DEMO DISC

Don't forget the new BEEBUGSOFT Demo disc launched in March. Check this month's codebreaker number and see if you have won free software of your choice. See the back cover for all the details.

PROGRAM CLASSIFICATION

All programs in the magazine, and on the magazine cassette/disc, are marked with the symbols shown below. An uncrossed symbol indicates full working, a single line through a symbol shows partial working for that configuration (normally some modifications will be required), and a cross through a symbol indicates a program that will not work on that type of system. There is a symbol for the B+ which includes the 128K version, and a symbol for the new Master series.

Basic I	I	Electron	◯
Basic II	II	Disc	💾
Tube	⊖	Cassette	📼
Model B+	✚	Master 128	M

News News News News News News Ne

NEW PENMEN

The Penman Plotter (see review in BEEBUG Vol.4 No.2) has been joined by some new companions capable of even more tricks. The Penman II series is capable of drawings up to AØ size (that's about 4ft x 3ft).

Looking very similar to the original Penman, the series II models can plot on semi-transparent materials such as tracing paper or drafting film and get their bearings from a black and white checkerboard placed underneath. On opaque materials the plotter is returned to the corners of the paper every so often. Prices are £395 for the IIB (A3 plots), £595 for the IID (A1 plots) and £695 for the IIE (AØ plots). Further details from Penman on Ø9Ø3-2Ø9Ø81.

SELF CONTROL

If the Beeb's humble User Port does not match up with your expectations then Claydale's in-out modules may be the answer. Up to eight modules can be plugged into the 1MHz bus giving a total of 64 inputs and 64 outputs all controllable from Basic or machine code with the routines supplied. Claydale also offer a range of other control modules to control stepper motors, servo units, and so on. Further details from Claydale on Ø6Ø3-868423.

CIVIL ACTION

The latest ROM from Vine Micros was written with the help of Civil Engineers at the University of Southampton. Called 'Matrix', the ROM simplifies operations on matrices from within Basic programs and will also solve linear simultaneous equations. Just for luck two other features - variable name swapping and a special four significant figure number print formatting - have also been added. Matrix costs £41.4Ø. Further details from Vine on Ø3Ø4-812276.

MASTER STROKES

First off the mark with such an announcement, Level 9 software reports that cassette versions of the following adventure games work perfectly on the Master. Colossal Adventure, Adventure Quest, Dungeon Adventure, Return to Eden, The Worm in Paradise, and Red Moon. Level 9 is on Ø494-26871. AMX too is quick to point out that Pagemaker (see the review in BEEBUG Vol.4 No.9) is available for the Master.

WORD GROUP

Members keen on word processors can now join a user group devoted to that application. The group publish a bi-monthly newsletter called 'Word Processing' which covers tutorials and hints in WP on the BBC, Amstrad and Atari micros. Subscription costs £6 a year. Further details from Word Processing, PO Box 67, Wolverhampton, West Midlands.

NEW BROM SWEEPS CLEAN

Clares has been busy recently with a plethora of updated goodies. Brom Plus now contains sections dealing with disc and RAM in addition to the original Basic utilities. Brom Plus costs £34.5Ø and the old Brom is reduced to £2Ø.

Fontwise Plus is the new version of Clares' Fontwise ROM with extra formatting commands, two extra fonts, faster printing, and it's View compatible. Fontwise Plus costs £2Ø and the old Fontwise £12. To the Beta-Accounts series Clares has added Statement and Stock control modules which integrate with the rest of the series (including Beta-Base reviewed in Vol.3 No.9).

Finally, Macrom is a macro assembler for all Beebs including the Master. Macrom will cope with the extended Op Codes used in the Master's processor and will assemble to and from memory or disc. Macrom costs £4Ø. Further details on all these from Clares on Ø6Ø6-48511.

BEEBUG SOFT FORUM

RomIt Exotic Titles

The !TITLE option in RomIt allows any RomIt-created Eprom to send a title message to the screen when the computer is turned on, and when Break is pressed. But the flexibility built into this option allows the title "message" to be any series of VDU commands, regardless of how complex these may be. This means that you can not only force a mode change, but draw any design, and print any text that you wish on Break and switch-on.

In fact you can create your exotic title from any graphics program that you may have. Suppose you have a 50 line program that draws a particular design on the screen, and you would like this to appear as the computer's start-up screen. Just add the following commands around your graphics program:

```
10*RAM
11*SPOOL !TITLEX
12VDU22,N  (Where N is
the graphics mode used)

..Main Graphics Program

1001*SPOOL
1002*RTITLE
```

Now run this program, and ignore the screen display. When it has finished, it will have created the required TITLE file in sideways Ram. Just press Break and the screen should be recreated. The file is now ready for romming; and of course you can use *FILE or *BLOCK to save it away to disc or cassette if you are not ready to rom it at this point.

Spellcheck III 50% Speed Increase

You may be able to increase the speed of Spellcheck quite dramatically. Acorn's later 1.2 DNFS (disc filing system) can give considerable speed advantages with Spellcheck (all versions). Tests with Spellcheck III (without Second Processor) gave timings of approximately 14 words per second when using the standard Acorn 0.9 DFS. This increased to an impressive 27 words per second with the 1.2 DNFS ROM. Even greater speeds are obtainable by using the 6502 Second Processor.

Exmon II. Machine Code for Beginners

The Beebugsoft customer support team are often asked the best way to get started with machine code. It may seem a daunting task, but experience has shown us that a good introductory book and a means of easily 'playing around' with machine code is the solution. We would recommend Assembly Language Programming for the BBC Microcomputer by Ian Birnabaum, and our own Exmon II. Normally the book costs £8.95 from Beebug Retail, and Exmon II £24 (to members). But members can now get this special 'beginners' package from BEEBUG Retail for £29 plus £1.00 p&p. This special offer closes 30th May 1986.

Beebugsoft Information Sheets

Several free information sheets are available from Customer Support giving additional help on various aspects of Beebugsoft software. These currently include:

Using Wordease with Hi-Wordwise

Using Wordease Disc Menu option on single drives

Sleuth Dual Screens explained

Using Paintbox II on the B+

Upgrading Magscan without a wordprocessor

Masterfile II and BREAK key

Spellcheck on the Opus DDOS

Spellcheck III words in Rom

Freelance Application Form

To obtain your sheet, send a stamped addressed envelope, to Beebugsoft, PO Box 50, Holywell Hill, St Albans, Herts AL1 3YS.

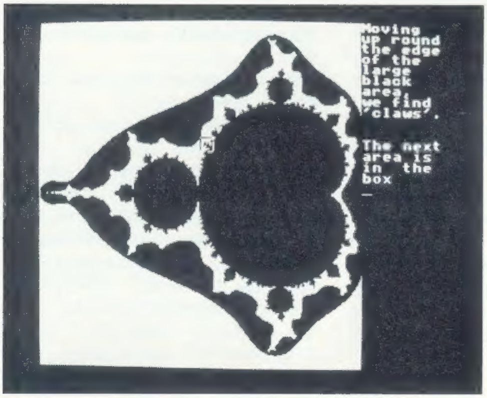
MANDELBROT GRAPHICS

Occasionally, some of the more obscure aspects of science acquire considerable popular interest. The so-called Mandelbrot Set is one such example, offering endless variations and stunning graphics that will keep you enthralled for hours. Kate Crennell has all the details.

Generating attractive screen displays is something that many BBC micro users like doing. Although the Beeb is very capable at displaying the results, it is often the subject matter that provides the stumbling block. This program sorts that out very simply. It produces an infinite number of incredible screens of swirling and exploding colour that can be re-drawn and magnified as you want, to show their infinite detail.

Although they appear almost random, the displays are based on the Mandelbrot Set - a mathematical set of numbers that you computer can generate and display on the screen in this fascinating way.

The program will plot the Mandelbrot Set according to the basic square law (W^2), and also the cubic (W^3) and quartic (W^4) laws. All three produce similar pictures with a central black area surrounded by coloured bands.


Moving up round the edge of the large black area, we find 'claws'.

The next area is in the box

You can 'zoom' in on sections of the plot and replot them. Smaller and smaller areas can be studied, revealing the infinite detail of the pattern, until you reach the limit of your Beeb's accuracy or your own patience.

When you have typed in the program and saved a copy, run it, following the on-screen instructions, using the square law, mode 2, and all the default values (answer 'Y' to the questions). This will draw the whole of the interesting area of the Mandelbrot Set.

This takes a fair time (around 25 minutes) so don't hold your breath. Fewer rows and columns mean less plotting time, but reduced detail. The maximum size in mode 2 is a square of side 128, and in mode 1 of side 256. It is usually possible to get a quick view of a plot using a side of 32. You are not limited to square plots; there is nothing to stop you trying distorted or rectangular ones.

When the plot is finished you can save the picture to disc or cassette and then zoom in on a small section. The 'box' feature (offered after the save option) allows you to determine easily the area to view next. First a line is drawn on the screen. Move the free end of this with the cursor keys to the bottom left of the area you want to view. Press L and then move to the top right corner and press R. Now that area will be enlarged to as near the size of the plotting window as is possible and drawn out on the screen.

The most exciting pictures are found along the edges of the black central area, particularly in its deeper cracks. Minature copies of the Mandelbrot Set can be found using more iterations. In their cracks, you will find other minature copies, and in their cracks.... The edges have a 'fractal' structure which means that the edge never gets any smoother no matter how high the magnification becomes.

You can alter the other parameters that make up the beauty of the finished plot. Firstly the resolution can be altered, but be warned that a mode 1 plot not only can be drawn in just four colours, but takes a VERY long time. The power of the equation used can also be changed. Again the higher power sets take longer to draw out. As well as determining the area to draw with the box, you can simply specify co-ordinates. However, it is probably a good idea to stick to the box method until you are more familiar with the set. To help you in this the X co-ordinate of the left hand side, the width, the Y co-ordinate of the bottom, and the height of the current plot are all printed to the right of the plot window, along with the power of the equation, and the current row number being plotted.

THE MATHEMATICS

Mathematically, the Mandelbrot Set is at the centre of a plane of complex numbers (Z) and can be drawn by using the iterative formula W=W^2 +Z where W is also a complex number. Non-mathematicians (and that includes the Beeb) can think of Z as a point (X,Y) on the screen. Let W be another point (U,V) and then calculate new values of U and V using the formulae:

```
    newU = (oldU)^2 -(oldV)^2 + X
    newV = 2 * (oldU) * (oldV) + Y
```

Keep on re-calculating new values of U and V until:

```
    (newU)^2 + (newV)^2 > 4
```

and count the number of iterations (the number of times the calculation is repeated). The number of iterations then determines the colour plotted at (X,Y) using a further set of rules.

The most interesting parameters to alter are probably the colours. The different colours are used to represent different groups of iteration numbers, rather like contours represent the boundries between groups of heights. Which colours are used for which groups and the extent of each group can be specified. The program defaults to 8 groups, but up to 15 can be used. Again these parameters are printed at the side of the plot.

At all times, while entering the parameters, pressing Escape will take you back to the previous screen, and the last set of parameters. Pressing Escape during the plotting will end the plot and enable you to save the screen or specify a new box.

SOME EXAMPLES

A few good areas of the Mandelbrot Set to look at are as follows:

With default iteration boundaries.

Left X	Low Y	width/height
-0.24	-1.16	0.26
-1.8	-0.07	0.07

With iteration boundaries set to 0,10,13, 19,25,30,35,40.

Left X	Low Y	width/height
-0.74	0.33	0.04
-0.19	1.02	0.06

DISC USERS

Because of the length of the program, disc users without shadow RAM should set PAGE to &1200 (type: PAGE=&1200 <Return>) before loading and running this program.

FURTHER INFORMATION

Those who wish to pursue this subject further may find the following references helpful:

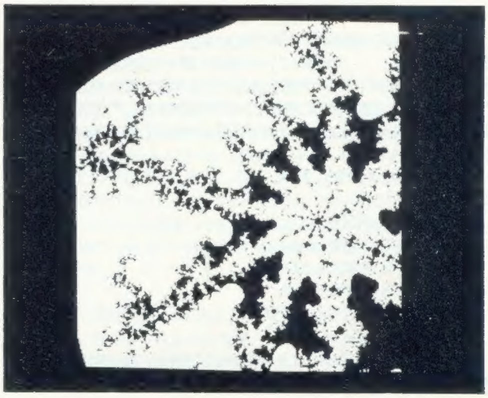
The Fractal Geometry of Nature, B.B.Mandelbrot, W.H.Freeman, 1982.

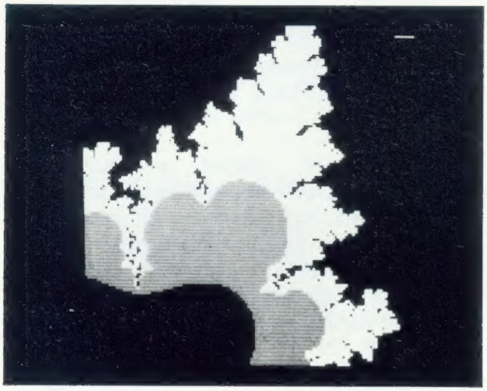
Scientific American, August 1985, Computer Recreations page 8.

PROGRAM NOTES

Line 290 saves the screen display to a file. You may want to add a screen compression routine here (such as was described in BEEBUG Vol.4 No.6) to save storage space. A corresponding de-compression routine will then be required to re-display the screen.

PROCmandel performs the algebraic iterations to calculate the colour, and plots each point in the area. Replacing





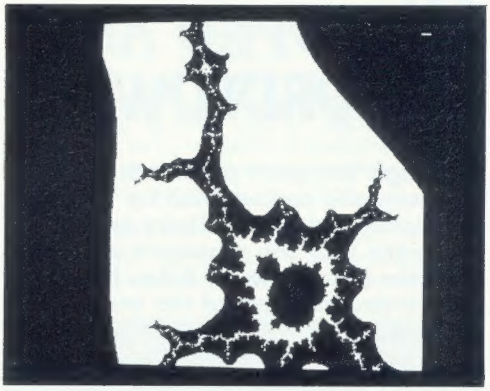
this procedure with one in machine code decreases the plotting time by a factor of three.

The main variables used are:

```
    M% - mode (1 or 2)
    T% - -1,0 or 1 for type of power law
         (2,3 or 4)
    C%+1 - number of colours in use
           (colours and iteration bounds
           are stored as bytes in page &C)
    X%+1 - number of columns to plot
    Y%+1 - number of rows
    DX, DY - size of a point
    XMIN, W, YMIN, H - definition of plot
           area
```

```
   10 REM PROGRAM MANDELBROT PLOT
   20 REM VERSION B0.2
   30 REM AUTHOR  Kate Crennell
   40 REM BEEBUG  MAY 1986
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 DIM KOL%(4),C$(7)
  110 ON ERROR GOTO 340
  120 C$(0)=CHR$135+"BLACK,  ":C$(1)=CHR
$129+"RED,    ":C$(2)=CHR$130+"GREEN, "
:C$(3)=CHR$131+"YELLOW,":C$(4)=CHR$132+
"BLUE,  ":C$(5)=CHR$133+"MAGENTA,":C$(6
)=CHR$134+"CYAN,   ":C$(7)=CHR$135+"WHIT
E, "
  130 VDU23,255,255,255,255,255,255,255,
255,255
  140 M%=0:MODE7
  150 E%=1:PROCmode
  160 E%=2:PROCrule
  170 E%=3:PROCarea
  180 MODE 7:E%=4:PROCkols
  190 MODE M%
  200 IFM%=1 FOR I%=0TO3:VDU19,I%,KOL%(I
%)MOD16;0;:NEXT
  210 VDU28,G%,31,H%,0:@%=4:PRINTT%+3:@%
=&20508:PRINTXMIN;:@%=&10308:PRINTW;:@%=
&20508:PRINTYMIN;:@%=&10308:PRINTH:@%=11
-M%*4:FORI%=0TOC%:PRINT?(&C10+I%);:COLOU
R?(&C00+I%):VDU255:COLOUR7:NEXT:@%=8/M%
  220 E%=6:VDU4,23;8202;0;0;0;28,G%,31,H
%,29
  230 PROCmandel(XMIN,YMIN,DX,DY)
  240 E%=5:VDU7,4,28,G%,31,H%,26:CLS
  250 PRINT"Save Y/N";:PROCQ:IF A$="Y"IN
PUT"Filename",F$:GOTO260 ELSE GOTO 310
  260 IFINSTR(F$,"""")<>0GOTO250 ELSE IF
 LEN(F$)>6GOTO250
  270 F$="P."+STR$(M%+2+2*T%)+F$
  280 IFM%=1 FOR I%=0TO3:?(I%+&7FFC)=KOL
%(I%)MOD16:NEXT
  290 $&C80="SAVE "+F$+" 3000 8000"
  300 U%=X%:V%=Y%:X%=&80:Y%=&C:CALL&FFF7
:X%=U%:Y%=V%
  310 PROCbox:IFA$="Y"MODE7:GOTO170
  320 CLS:PRINT"MORE Y/N";:PROCQ:IF A$="
Y"GOTO180
  330 MODE7:END
  340 IFE%<>5 OR ERR<190 GOTO360
  350 PRINT"FILING ERROR ";ERR:GOTO250
  360 IF ERR<>17 MODE7:REPORT:PRINT;" AT
 LINE ";ERL:END
  370 ON E% GOTO 380,150,160,170,180,240
  380 MODE 7:END
  390 :
 1000 DEFPROCkols
 1010 J%=0
 1020 CLS:PRINT''"NUMBER COLOUR  LOWER B
OUND"
 1030 @%=6:FORI%=0TOC%:K%=?(&C00+I%):IFM
%=1 K%=KOL%(K%)
 1040 PRINTI%+1;:IFK%>0PRINT"   "+CHR$(1
28+K%)+CHR$255+CHR$135,?(&C10+I%) ELSE P
RINTSPC(6),?(&C10+I%)
 1050 NEXT:PRINT'"These are the colours
to be plotted for each new boundary of i
terations."'
 1060 IFJ%=C%+1 GOTO1110 ELSE:PRINT''"Num
ber of boundaries OK (Y/N)";:PROCQ
 1070 IFA$="Y" J%=C%+1:GOTO1110
 1080 PRINT"How many boundaries (2-15) "
;:J%=GET-48:PRINT;J%;:IFJ%<1 OR J%>9 GOT
O1080
 1090 IFJ%=1:K%=GET-48:PRINT;K%;:J%=10+K%
 1095 IFJ%<>C%+1 PROCbnd
 1100 IFJ%<2 OR J%>15 GOTO1070 ELSE C%=J
%-1:GOTO1020
 1110 PRINT''"Which colour/bound to chang
e?"'"(RETURN if none)";:K%=GET-48:PRINT;
K%;:IF K%=0 OR K%=-35 ENDPROC
 1120 IFK%=1 AND J%>9 I%=GET-48:IF I%>-3
5 K%=10+I%:PRINT;I%;
 1130 IFK%>J% GOTO1110
 1140 IFM%=2PRINT'"Colours are:":FORI%=0
TO7:PRINT;C$(I%)+" ";:NEXT:GOTO1200
```

```
 115Ø PRINT'"Background colour is "+C$(K
OL%(Ø)MOD8)'"Printing will be in "+C$(KO
L%(3)MOD8)
 116Ø PRINT"Primary colours available:":
FORI%=ØTO3:IFKOL%(I%)<16 PRINT;C$(KOL%(I
%))+" ";
 117Ø NEXT:PRINT'"Other possible colours
:":FORI%=ØTO7:F%=Ø:FOR N%=ØTO3:IFKOL%(N%
)=I% F%=1
 118Ø NEXT:IFF%=Ø PRINT;C$(I%);SPC1;
 119Ø NEXT
 12ØØ I%=1Ø:INPUT'"Which colour",A$:IFA$
=""GOTO133Ø ELSE A$=LEFT$(A$,3)
 121Ø FORN%=ØTO7:IFA$=MID$(C$(N%),2,3) I
%=N%
 122Ø NEXT
 123Ø IFI%=1ØGOTO12ØØ
 124Ø IFM%=2 ?(&BFF+K%)=I%:GOTO132Ø
 125Ø F%=Ø:FORN%=ØTO3:IF I%=KOL%(N%) ?(&
BFF+K%)=N%:F%=-1
 126Ø IFF%=Ø AND KOL%(N%)=16 F%=N%+1
 127Ø NEXT:IF F%>Ø KOL%(F%-1)=I%:?(&BFF+
K%)=F%-1:GOTO 132Ø
 128Ø IFF%=Ø KOL%(?(&BFF+K%))=I%:GOTO 13
2Ø
 129Ø FOR I%=ØTO3:F%=Ø:FORN%=ØTOC%:IF?(&
CØØ+N%)=I% F%=1
 13ØØ NEXT:IF F%=Ø KOL%(I%)=16
 131Ø NEXT
 132Ø IFK%=1 GOTO1Ø2Ø
 133Ø INPUT"What new lower iteration bou
nd for this colour",L%:IFL%<Ø GOTO133Ø
 134Ø IFL%=ØGOTO1Ø2Ø
 135Ø IFL%<?(&CØE+K%)+1 GOTO139Ø
 136Ø IFL%>255 GOTO139Ø
 137Ø IFK%<J% AND L%>?(&C1Ø+K%)-1 GOTO13
9Ø
 138Ø ?(&CØF+K%)=L%:GOTO1Ø2Ø
 139Ø PRINT"New range incompatible with
old":GOTO133Ø
 14ØØ :
 141Ø DEFPROCarea
 142Ø CLS:@%=&2Ø7ØA:PRINT''"Left X bound
="XMIN;", width="W'"Low Y bound="YMIN;",
 height="H':PROCask:IFA$="Y" GOTO147Ø
 143Ø INPUT'"New left X",XMIN
 144Ø INPUT"New width",W:IFW=ØGOTO144Ø
 145Ø INPUT"New bottom Y",YMIN
 146Ø INPUT"New height",H:IFH=ØGOTO146Ø
 147Ø L%=256/M%:R=ABS(H/W):IFR>1 Y%=L%-1
:X%=Y%/R+Ø.5 ELSE X%=L%-1:Y%=X%*R+Ø.5
 148Ø @%=4:PRINT'"The plot will have ";X
%+1;" columns"'"and ";Y%+1;" rows."'':PR
OCask:IFA$="Y" GOTO151Ø
 149Ø PRINT'"New number of columns ( <";
L%+1" )";:INPUT,I%:IFI%<1 OR I%>L% PRINT
" Illegal value":GOTO149Ø ELSE X%=I%-1
 15ØØ PRINT"New number of rows ( <";L%+1
" )";:INPUT,I%:IFI%<1 OR I%>L% PRINT" Il
legal value":GOTO149Ø ELSE Y%=I%-1
```



```
 151Ø DX=W/X%:DY=H/Y%:ENDPROC
 152Ø :
 153Ø DEFPROCask
 154Ø *FX2Ø2,4Ø
 155Ø PRINT;" Is this OK (Y/N)";:PROCQ:E
NDPROC
 156Ø DEF PROCQ:PRINT;"?";
 157Ø  A$=CHR$(GET  AND 223):IFA$<>"Y" AND
A$<>"N" GOTO 157Ø
 158Ø PRINT;A$:ENDPROC
 159Ø :
 16ØØ DEFPROCmode
 161Ø T%=-1:@%=1Ø
 162Ø CLS:PRINTTAB(7,1)CHR$134"The Mande
lbrot Set"''
 163Ø PRINT"There are 2 available plot m
odes:"''"1)"+CHR$131+"High"+CHR$135+"res
olution (256 lines) and"'"four colours,"
''"2) Low resolution (128 lines) and"'CH
R$131+"eight"+CHR$135+"colours."'
 164Ø IFM%=Ø PRINT'"Which mode would you
 like (1 or 2)? ";:M%=GET-48 ELSE GOTO 1
66Ø
 165Ø IFM%<1 OR M%>2 M%=Ø:GOTO 164Ø ELSE
 PRINT;M%:GOTO 167Ø
 166Ø PRINT"Plot mode will be"+CHR$134;M
%;CHR$135;:PROCask:IF A$="N" M%=3-M%:GOT
O166Ø
 167Ø ON M% GOTO 168Ø,169Ø
 168Ø G%=32:H%=39:C%=3:FOR I%=ØTO15:?(I%
+&C1Ø)=4*I%:?(I%+&CØØ)=(I%+1)MOD4:NEXT:K
OL%(Ø)=Ø:KOL%(1)=4:KOL%(2)=6:KOL%(3)=7:E
NDPROC
 169Ø G%=16:H%=19:C%=7:FOR I%=ØTO15:?(I%
+&C1Ø)=2*I%:NEXT:!&CØØ=&Ø2Ø5Ø1Ø4:!&CØ4=&
ØØØ7Ø3Ø6:!&CØ8=&Ø2Ø5Ø1Ø4:!&CØC=&Ø8Ø7Ø3Ø6
:ENDPROC
 17ØØ :
 171Ø DEFPROCrule
 172Ø CLS:@%=1Ø:PRINT''"The Mandelbrot f
ormula used will be:"''"    Z^";T%+3;" +
 C => Z'"'':PROCask:IFA$="Y" GOTO175Ø
```

→ 12

WORDPRO AND WORDPOWER

Although the word processor market is dominated on the Beeb by Wordwise and View, others are available. Can the minnows compete with the giants? Geoff Bains has been trying out two of the better alternatives.

Product	**: Wordpower**
Supplier	**: Ian Copestake**
Address	**: 23 Connaught Crescent, Brookwood, Woking GU24 0AN.**
Phone	**: 04867-4755**
Price	**: £39**

Product	**: ProWord**
Supplier	**: T. Hall**
Address	**: 31 Wetherfield Rd, Noctorum, Wirral, Merseyside L43 9YF.**
Phone	**: 051-652-7560**
Price	**: £27.50**

WORDPOWER

Wordpower is supplied on both ROM and disc, and both must be present (at least initially) for the program to work. Wordpower is compatible with the B+, Master, Tube and Electron.

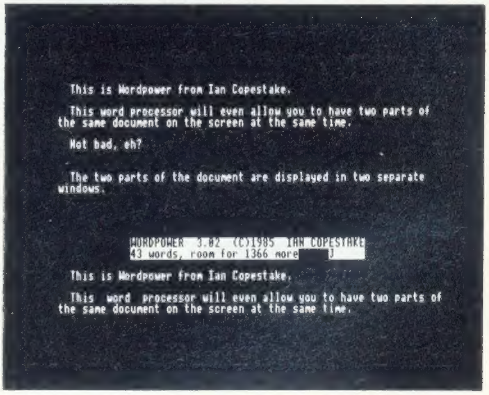
Wordpower is another 'what you see is what you get' (WYSIWYG) word processor that owes a lot to both View and Wordwise. Like View, most commands are function key based and the text formatting is controlled by way of rulers inserted in the text. Lines can be up to 250 characters long with up to 28 Tab stops.

Text is entered in any mode, so if the line length chosen is greater than the chosen mode can display, the line is split over several screen lines in the same way as that used in the preview mode of Wordwise. For a very strange effect a function key enables you to 'switch off' any of the 'sliced' lines on the screen in an attempt to make the display look more like the printed result and more readable. This isn't very effective. There is no sideways scrolling over long lines, even to the limited extent offered by View.

Like Wordwise, the rest of the control over text is placed in the hands of embedded codes. These can appear anywhere in the document and affect such features as line spacing, heading and footing spaces and text, page numbering, and so on.

The most unusual aspect of Wordpower is the way in which it treats long documents, too long for memory. Large files of text on disc are edited a section at a time. However, it doesn't automatically save and load chunks of text, as you need them, like a 'real' word processor, but requires you to do the work by 'yanking' the next piece of text into the machine as and when you need it. This takes some time to get used to, and although a useful facility, to be honest, I prefer to deal with several short files as I choose.

A lot of the workings of Wordpower are tedious. To save just a section of your document you have to move around the text specifying the start and end of the section to be saved and hit the Y key twice in response to prompts. Admittedly Wordpower does its best to help, and you can use the Ctrl-Q (Quick Save) option,


```
  This is Wordpower from Ian Copestake.

  This word processor will even allow you to have two parts of
the same document on the screen at the same time.

  Not bad, eh?

  The two parts of the document are displayed in two separate
windows.

            WORDPOWER  3.02  (C)1985  IAN COPESTAKE
            43 words, room for 1366 more     J

  This is Wordpower from Ian Copestake.

  This  word  processor will even allow you to have two parts of
the same document on the screen at the same time.
```

before the save command itself, to save all your text, but that is no real answer. The most commonly used save has to be to save the entire document. So why not have a single command devoted to doing just this?

In one aspect Wordpower is outstanding. It can actually work on two documents at once. One document can be printed while you are editing another. Even more impressively, the screen can be split into two separate windows, one above the other, each with its own cursor position and other options, and each displaying a different part of the same document or even, with a little judicious fiddling, two separate files (though they are not strictly speaking separate when in the machine).

A lot of thought and work has gone into Wordpower to make it as powerful as possible. However, the end result is unfortunately a little unwieldy. Maybe this is inevitable given the features included, but I doubt it.

PROWORD

ProWord is supplied on a single ROM. There is a HiWord version for users of the 6502 second processor and this is available on a separate ROM. ProWord consists of three separate parts. Firstly, there is the editor. This is the actual text manipulation section of the program.

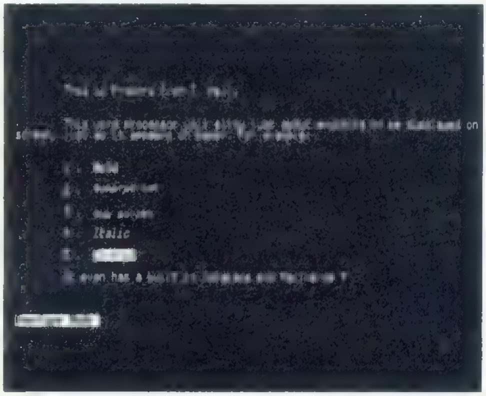
ProWord is an excellent example of a WYSIWYG word processor. Text is entered on an 80 column screen. all the formatting is provided by function key commands (on their own and along with Shift, Ctrl, and Ctrl-Shift). There are no rulers to contend with, but invisible commands are inserted into the text wherever the formatting function keys are pressed.

When a formatting function key is pressed the bottom of the screen turns into a ruler of sorts and the margins, tab positions or whatever are set with the cursor keys. The drawback with the system is that when text is actually typed in, it isn't formatted. Another function key must be pressed to format each paragraph.

Despite this drawback ProWord is still extremely powerful. The text on the screen is rather spindly. This is because it is not bold type. Pressing another function key allows you to select a font. These are fine, bold, superscript, subscript, underline and inverse. These different fonts are displayed on the screen, just as they appear on paper.

The whole of ProWord is steeped in features. Even the humble search and replace has many options added that you would not expect to find on a word processor for the Beeb. The replace routine can distinguish between whole words or just sections; it can operate backwards or forwards through a document; it will ask on each find if replacement is wanted; it will reformat a paragraph after replacement; it will even replace just the first specified number of finds.

All these options are selected from a menu system as are all the facilities of this word processor not directly concerned



with editing. Such versatility is of course impressive. However, it does mean that using this word processor can become a little cumbersome.

The other facilities offered by ProWord include saving and loading text from disc or cassette and spooling to disc or cassette a formatted ASCII file. Sadly there is no facility to save sections of a document.

The second section of ProWord is the printer driver. This is similar to the View printer driver and is used to specify all the codes used by your particular printer to perform such tasks as underlining, emphasized type, and so on.

Unlike View (versions before 3.00), the printer driver is included in the ROM. Defining the codes is a simple question and answer process. ProWord asks if each of the features are available on your printer and if they are, then what the codes are to initiate them.

At the same time as the printer codes

are being defined you can also define which disc drives are to be used for what. The codes you define can be saved to disc or cassette and loaded in at a later date so you don't have to go through the whole process again.

The third section of this word processor is the 'database'. This option is only available to disc users of ProWord. It is a simple cardbox type database which can handle several hundred records of up to seven fields of up to 79 characters. The records may be interrogated in a simple fashion and printed out.

However, the real use for the database is as a mail merging facility. By incorporating @1 to @7 in your document, the relevant fields from the database records are incorporated into the text. There is even a facility to reformat the text automatically (!1 to !7 are used) after the insertion of the data field. This quite advanced facility enables you, say, to write personal sounding letters to a large number of people with the minimum of effort.

CONCLUSIONS

ProWord is certainly feature-filled. Like Wordpower, a lot of effort has gone into packing an awful lot of word processor into the humble Beeb. However, both packages do suffer a little from over complication in use. It takes a long while to become anything like familiar with these word processors. Although both packages are way ahead of the likes of Wordwise in terms of facilities, I still return to that old workhorse with a sense of relief. After all, a word processor is meant to make the act of writing easier. If using a word processor makes life more difficult then little has been gained.

←9

Mandelbrot Graphics

```
 1730 PRINT'"Enter new exponent (2-4) ";
:I%=GET-48:PRINT;I%:IF I%<2 OR I%>4PRINT
"Illegal value":GOTO1730
 1740 T%=I%-3:GOTO1720
 1750 ON T%+2 GOTO 1760,1770,1780
 1760 XMIN=-2:YMIN=-1.25:W=2.5:H=W:ENDPR
OC
 1770 XMIN=-1.5:YMIN=-1.5:W=3:H=3:ENDPRO
C
 1780 XMIN=-1.5:YMIN=-1.25:W=2.5:H=W:END
PROC
 1790 :
 1800 DEFPROCbox
 1810 CLS:PRINT"BOX Y/N";:PROCQ:IFA$="N"
ENDPROC
 1820 CLS:S%=4*M%:U%=0:V%=0:A%=640:B%=51
2:GCOL3,3:MOVEU%,V%:DRAWA%,B%
 1830 *FX4,1
 1840 *FX12,4
 1850 A$=GET$:REPEAT:PROCbx:UNTIL A$="Q"
 1860 *FX4,0
 1870 *FX12,0
 1880 CLS:PROCask:MOVEU%,V%:DRAWU%,B%:DR
AWA%,B%:DRAWA%,V%:DRAWU%,V%:IFA$="N"GOTO
1820
 1890 CLS:PRINT"Draw new one";:PROCQ:IFA
$="N"ENDPROC
 1900 XMIN=XMIN+DX*U%/S%:YMIN=YMIN+DY*V%
/S%:W=(A%-U%)*DX/S%:H=(B%-V%)*DY/S%:ENDP
ROC
 1910 ENDPROC
 1920 :
 1930 DEFPROCbx
 1940 IF INSTR("LR",A$)=0 $=GET$:ENDPROC
 1950 MOVEU%,V%:DRAWA%,B%:IFA$="L"U%=A%:
V%=B%ELSE IFA$="R"MOVEU%,V%:DRAWU%,B%:DR
AWA%,B%:DRAWA%,V%:DRAWU%,V%:A$="Q":ENDPR
OC
 1960 IFA$=CHR$136 A%=A%-S% ELSEIFA$=CHR
$137 A%=A%+S% ELSEIFA$=CHR$139 B%=B%+S%
ELSEIFA$=CHR$138B%=B%-S%
 1970 MOVEU%,V%:DRAWA%,B%:A$=GET$:ENDPRO
C
 1980 :
 1990 DEFPROCmandel(XMIN,YMIN,DX,DY):P%=
C%?&C10:L%=M%*4
 2000 Y=YMIN-DY:FORV%=0TOY%:PRINTV%:Y=Y+
DY:X=XMIN-DX:FORU%=0TOX%:X=X+DX:U=X:V=Y:
N%=0
 2010 ONT%+2GOTO2020,2030,2040
 2020 A=U*U:B=V*V:IFA+B>4GOTO2050ELSEN%=
N%+1:IFN%=P%GOTO2050ELSEV=Y+2*U*V:U=X+A-
B:GOTO2020
 2030 A=U*U:B=V*V:IFA+B>2GOTO2050ELSEN%=
N%+1:IFN%=P%GOTO2050ELSEV=Y+V*(3*A-B):U=
X+U*(A-3*B):GOTO2030
 2040 A=U*U:B=V*V:IFA+B>2GOTO2050ELSEN%=
N%+1:IFN%=P%GOTO2050ELSEV=Y+4*U*V*(A-B):
U=X+A*A+B*B-6*A*B:GOTO2040
 2050 J%=C%?&C00:FORI%=C%TO1STEP-1:IFN%<
I%?&C10 J%=I%?&BFF
 2060 NEXT:GCOL0,J%:PLOT69,U%*L%,V%*L%:I
FM%=2PLOT69,U%*L%,V%*L%+4
 2070 NEXT,:ENDPROC
```

THE BARRY BOX

The Barry Box is one of those devices that just has to be heard to be believed. The device provides all that is needed to store and reproduce the most realistic speech and sound. Geoff Bains has been exercising his vocal chords.

Product : Barry Box
Supplier : BML Electronics,
Unit 24, Larch Grove,
Bletchley, MK2 2LL.
0908-640805
Price : £79.95

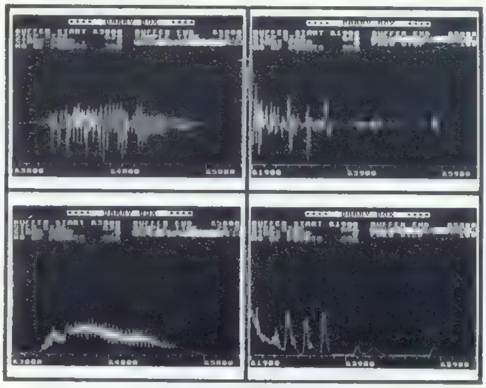
Paul Hardcastle's song of last year not only brought the number 19 to the public's eye but also the idea of digital sound sampling. That record was made with equipment that converts a noise into digital values that may be manipulated by a computer. Now the Barry Box gives BBC micro owners the same facilities.

The Barry Box itself is a small black plastic case with cables to connect it to the Beeb's 1MHz bus and auxiliary power supply. A microphone (supplied) plugs into the box and an extension speaker or amplifier can be connected too, if the Beeb's own sound system is not to your taste. Of course, the software to run it all is supplied on ROM.

The hardware consists mainly of fast ADC and DAC chips. The ADC (Analogue to Digital Converter) samples the input from the microphone several thousand times a second, and converts this into digital form. The Beeb can store and manipulate these values, and the DAC (Digital to Analogue Converter) can turn them back into sound.

The Barry Box can be used in two modes. In Direct mode a *BAR. command puts the Beeb under the command of the software and gives you several options, all selectable with the function keys.

Sample a sound (key triggered)
Sample a sound (sound triggered)
Playback sound
Playback sound backwards
Load or Save sampled sound to tape,
disc or sideways RAM


Sound Graphs

Display waveform
Display frequency analysis
Display amplitude envelope
Frequency smoothing
Amplitude smoothing

The frequency analysis is particularly impressive. Complicated mathematical calculations are used to reduce your sampled sound into a graph of frequency against amplitude. The sampling rate can be altered allowing your own choice of compromise between quality of sound (from good to appalling) and length of sound sample (up to about ten seconds).

Such facilities make the Barry Box an obvious choice for demonstrating the nature of sound in schools. However, the Barry Box isn't just good for serious stuff; it's great fun too. If you thought that the sound of your own voice was entertaining then try it backwards.

In Program mode all these facilities of the Barry Box are available to a Basic program with a series of * commands. These commands are given names to tie in with the Direct mode (e.g. playback is initiated by pressing f2 in Direct mode and with *BAR2 in Program mode), making them easy to remember. Sampled sounds can be easily strung together or chopped around from Basic. The short but comprehensive manual gives a few example programs. A simple 10 line Basic program produces a reasonable facsimile of the 'N-N-N-Nineteen' effect. Whether for study or fun, the Barry Box, though expensive, is an impressive piece of equipment.

WINDOWS AND ICONS
(Part 3)

Alex Kang brings this series to a climax with a complete application of the USI routines previously described. The Telephone Book shows the tremendous power and user-friendliness of this approach to writing applications and utilities.

The previous two articles in this series described the USI utilities for user-friendly screen displays, and provided a simple demonstration of their use. Here, in the last article of the series, is a 'phone book' program which demonstrates another possible application of the utilities. The program displays a telephone book on the screen in which you may enter and retrieve names and telephone numbers just as you would in an ordinary telephone book.

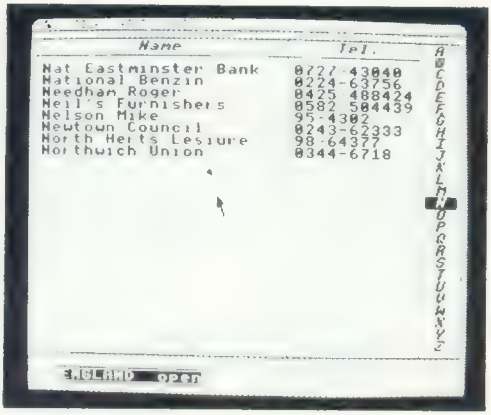
TYPING IN THE PROGRAM

As this program makes use of random access filing, it is for disc users only. However, the principles behind the application of the USI utilities apply also to cassette users.

Type in listing 1 and save it to a fresh disc containing just a copy of the assembled USI routines (M.USI) from part 1. This program sets up the screen display and special character set. Save this program as "B.PHONE1", and then type in listing 2 (Basic I users should change all occurrences of OPENUP to OPENIN if typing in the magazine listing). Save listing 2 as "B.PHONE2". To use the program, just CHAIN "B.PHONE1". This assumes that your USI utilities code is saved as "M.USI". On the Master 128 PAGE should first be set to &1900 (type PAGE=&1900 <Return>) to provide the space for the USI routines.

USING THE PROGRAM

There are four general options: "system manager" (represented by the 'mouse' symbol top left), "file", "edit" and "quit", all of which display a pop-up window when selected. The program operates on files (phone books) under directory "T", so make sure there



```
Name                            Tel.
Nat Eastminster Bank            0727 43040
National Benzin                 0224-63756
Needham Roger                   0425 488424
Neil's Furnishers               0582 504439
Nelson Mike                     95-4302
Newtown Council                 0243-62333
North Herts Lesiure             98 64377
Northwich Union                 0344-6718
```

are no other files with directory "T" on the same disc, apart from those you create using this program.

Of course, the first time you use the program, you will not have any data files (phone books) so you must create one. You do this by selecting the "Create file" option under the general option "file". Type in the filename of the phone book you wish to create. Pressing Return without entering a filename just closes the pop-up window.

The "Load file" and "Delete file" options will display up to 10 files, or phone books, so make sure you do not exceed this number. When using these options, just select the file you wish to load or delete. However, pressing the space bar without pointing to any file will just close the window.

Once a file has been created or loaded, it will, by default, display page "A". Selecting any of the letters down the right hand side of the screen will result in the contents of that page being displayed. Use the "edit" option to write entries, delete the page, or sort the entries on the page. When writing, you simply select the line you wish to write to using the arrow pointer. You may select either the name or number fields. Selecting the former enables you to write to both name and number fields, while selecting the latter will only write to the number field. If you wish to leave the entry unchanged, just press Return. To delete the entry, press Space followed by Return. These operations are only carried out on the current entry field selected, which is shown highlighted. Once you have finished editing the page, select the "Quit write" box at the bottom of the screen.

Select the "Sort entries" option to sort out the entries on the page. You may choose to save the sorted entries onto the file or not. Similarly, the "Delete page" option will delete all entries on the page upon confirmation. This is useful for 'blanking' the pages of a newly-created file if necessary. The "Quit" option will exit from the program upon confirmation.

The little picture of the mouse on the top left hand corner of the screen acts as the system manager - you may use this to change the pointer speed and screen colours.

CUSTOMISING THE PROGRAM

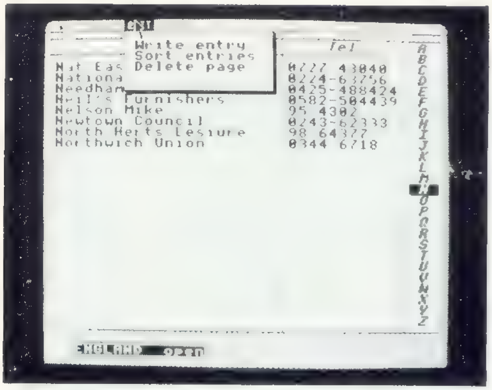
Each page of a phone book holds 24 entries. However, some letters are not too commonly used e.g. "X","Q" or "Z". You may combine letters in this case, e.g."XYZ" or "CD" etc. by altering the data statement (line 1060) in listing 1. Another idea is to reserve a page for your most frequently used numbers, by denoting it, say, with an asterisk "*" instead of with a letter. Whatever you choose, you must make sure that there are still 26 page labels, though some could just be blank of course.

The phone book provides both a most useful application program and an excellent example of the power and user-friendliness that can be achieved with the USI utilities. Careful study of this program will help you in writing your own USI applications.

```
   10 REM Telephone Book Screen Setup
   20 REM Listing 1, Version B1.0
   30 REM Author Alex Kang
   40 REM Beebug May 1986
   50 REM Program subject to copyright
  100 HIMEM=&4D00:*LO."M.USI"
  110 PROCin:PROCatoz:PROCch:PROCsm("fil
e   edit   quit",128,1023)
  120 CHAIN"B.PHONE2"
 1000 DEFPROCin:VDU22,4,19,0,7;0;19,1,4;
0;23;8202;0;0;0;23,252,240,192,128,128,0
,0,0,0,23,253,15,3,1,1,0,0,0,0,23,254,36
,114,226,25,61,30,60,0,23,255,17,68,17,6
8,17,68,17,68,31,0,0,252,254,31,39,0,253
 1010 ?&D0=2:FORj%=1TO31:PRINTTAB(0,j%)S
TRING$(40,CHR$255);:NEXT:?&D0=0:VDU25,4,
0;991;25,5,1279;991;24,80;16;528;80;16,2
6,23,255,255,255,255,255,255,255,255,255
 1020 VDU24,16;111;1263;967;16,26,25,4,1
6;967;25,5,1263;967;25,5,1263;111;25,5,1
6;111;25,5,16;967;25,4,16;919;25,5,1136;
919;25,4,1136;967;25,5,1136;111;25,4,752
;967;25,5,752;111;
 1030 V%=0:P%=0:E%=0:H%=0:S%=0:ENDPROC
 1050 DEFPROCatoz:FORc%=1TO26:READc$:PRO
Cital(c$,37,c%+1):NEXT:PROCital("Name",1
0,2):PROCital("Tel.",28,2):ENDPROC
 1060 DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,O
,P,Q,R,S,T,U,V,W,X,Y,Z
 1080 DEFPROCital(A$,x%,y%):VDU31,x%,y%:
italic=&178F:FORi%=1TOLENA$:?&80=ASC(MID
$(A$,i%,1)):CALL italic:VDU23,254:FORj%=
1TO8:VDUj%?&80:NEXT:VDU254:NEXT:ENDPROC
 1100 DEFPROCsm(A$,gx%,gy%):VDU5:FORi%=1
TOLENA$:a%=(ASC(MID$(A$,i%,1)) AND 223):
a%=a%-159*(a%>64 AND a%<91):MOVEgx%,gy%:
VDUa%:gx%=gx%+20-4*(a%=77 OR a%=87):NEXT
:VDU4:ENDPROC
 1120 DEFPROCch:FORi%=0TO25:VDU23,224+i%
,0:FORj%=1TO6:READk%:VDUk%:NEXT:VDU0:NEX
T:FORi%=250TO253:VDU23,i%:FORj%=1TO8:REA
Dk%:VDUk%:NEXT,:ENDPROC
 1130 DATA96,144,144,240,144,144,96,144,
224,144,144,224,112,128,128,128,128,112,
224,144,144,144,144,224,96,144,144,224,1
28,112,240,128,224,128,128,128,96,144,12
8,176,144,96,144,144,240,144,144,144,224
,64,64,64,64,224,16,16,16,16,144,96
 1140 DATA144,160,192,160,144,144,128,12
8,128,128,128,240,112,168,168,168,168,16
8,96,144,144,144,144,144,96,144,144,144,
144,96,96,144,144,224,128,128,96,144,144
,208,160,80,96,144,144,224,144,144,112,1
28,240,16,16,224,224,64,64,64,64,64
 1150 DATA144,144,144,144,144,96,144,144
,144,144,160,64,136,168,168,168,168,112,
144,144,96,96,144,144,144,144,144,112,16
,224,240,16,32,64,128,240
 1160 DATA255,128,128,135,159,184,186,13
1,240,8,4,226,249,29,93,193,142,157,189,
190,159,128,128,255,113,185,189,125,249,
1,1,255
```

```
 10 REM TELEPHONE BOOK
 20 REM Listing 2, Version B1.0
 30 REM Author Alex Kang
 40 REM BEEBUG May 1986
 50 REM Program subject to copyright
 60 :
100 DIM str 50,F$(10),N$(24),T$(24)
110 PROCinit:PROCld
120 REPEAT:IF H%=0 PROCs(0)
```

```
  130 REPEAT:RESTORE210:CALL pointer:PRO
Csel(2,5):UNTILS%>Ø:M%=S%
  140 IF M%=5 AND H%>Ø PROCaz
  150 IF M%<>5:RESTORE220:FORi%=1TOM%:RE
ADx%,y%,z%:NEXT:PROCfb(x%,y%,z%):IF M%<4
:IF FNsub>Ø PROCchk
  160 IF M%=4 PROCquit
  170 IF M%<>5 AND E%=Ø:PROCfb(x%,y%,z%)
  180 S%=Ø:UNTIL E%=1:*DIR $
  190 MODE7:IF H%>Ø CLOSE#H%
  200 END
  210 DATA 1,1,Ø,Ø,1,4,6,Ø,Ø,2,8,10,Ø,Ø,
3,13,14,Ø,Ø,4,36,38,2,27,5
  220 DATA 1,Ø,1,3,Ø,4,8,Ø,3,12,Ø,4
  230 :
 1ØØØ DEFPROCinit:T%=8Ø:CF$=""
 1Ø1Ø open=&131D:close=&134F:pointer=&15
54:input=&166C:str=&16C7:curoff=&16F9:cu
ron=&1711:qubox=&1728:oscli=&1787:*DIR T
 1Ø2Ø ?&7Ø=2Ø:?&71=16:fcon$=CHR$25Ø+CHR$
251+CHR$8+CHR$8+CHR$1Ø+CHR$252+CHR$253:P
%=Ø:V%=Ø:f%=Ø:S$=STRING$(22," "):PROCqb(
3,3Ø,STRING$(13," ")):ENDPROC
 1Ø3Ø :
 1Ø4Ø DEFPROCs(r%):VDU26:?&D3=255:s$=FNp
ad(9,CF$)+"open":IF r%=Ø s$="No files op
en"
 1Ø5Ø PRINTTAB(3,3Ø)s$:?&D3=Ø:ENDPROC
 1Ø6Ø :
 1Ø7Ø DEFPROCchk
 1Ø8Ø IF M%=1 AND S%=1 PROCps
 1Ø9Ø IF M%=1 AND S%=2 PROCcol(Ø)
 11ØØ IF M%=1 AND S%=3 PROCcol(1)
 111Ø IF M%=2 AND S%=1 PROCld
 112Ø IF M%=2 AND S%=2 PROCdf
 113Ø IF M%=2 AND S%=3 PROCcf
 114Ø IF H%=Ø ENDPROC
 115Ø IF M%=3 AND S%=1 PROCwrite
 116Ø IF M%=3 AND S%=2 PROCsort
 117Ø IF M%=3 AND S%=3 PROCdp
 118Ø ENDPROC
 119Ø :
 12ØØ DEFPROCquit:IF FNconf(12,"QUIT PRO
GRAM"," quit ")=1 E%=1
 121Ø PROCcl:ENDPROC
 122Ø :
 123Ø DEFPROCld
 124Ø PROCfl:IF nf%>Ø f%=FNic
 125Ø IF (nf%=Ø OR f%=Ø) PROCcl:ENDPROC
 126Ø CF$=F$(f%):fn$="T."+F$(f%)
 127Ø IF H%>Ø CLOSE#H%:PROCfb(36,V%,3)
 128Ø H%=OPENUP fn$:PROCcl:PROCfb(36,2,3
):P%=1:PROCsp:V%=2:S%=Ø:PROCs(1):ENDPROC
 129Ø :
 13ØØ DEFPROCfl:PROCgf:B%=6-4*(nf%>5):PR
OCop(3,B%,33,1,2):IF nf%=Ø VDU7:PRINT"NO
 PHONE BOOKS ON DISC."'"Press <RETURN>..
.":REPEATUNTILINKEY-74:ENDPROC
 131Ø IF nf%>5 PROCpfi(1,5,Ø,895):PROCpf
i(6,nf%,4,767) ELSE PROCpfi(1,nf%,Ø,895)
 132Ø ENDPROC
 133Ø :
 134Ø DEFPROCgf:?&8Ø=Ø:!&81=&AØØ:!&85=31
:!&89=Ø:X%=&8Ø:Y%=Ø:A%=8:CALL &FFD1
 135Ø nf%=31-?&85:IF nf%>1Ø nf%=1Ø
 136Ø IF nf%=Ø ENDPROC
 137Ø FORj%=1TOnf%:F$(j%)="":FORi%=1TO7:
A=i%?(&AØØ+(j%-1)*8):F$(j%)=F$(j%)+CHR$(
A):NEXT,:ENDPROC
 138Ø :
 139Ø DEFPROCop(a%,b%,c%,d%,w%)
 14ØØ A%=a%:B%=b%:C%=c%:D%=d%:W%=w%
 141Ø CALL open,A%,B%,C%,D%,W%:ENDPROC
 142Ø :
 143Ø DEFPROCcl
 144Ø CALL close,A%,B%,C%,D%:ENDPROC
 145Ø :
 146Ø DEFPROCqb(x%,y%,q$)
 147Ø $str=q$:CALL qubox,x%,y%:ENDPROC
 148Ø :
 149Ø DEFPROCsmall(A$,gx%,gy%):VDU5:FORi
%=1TOLENA$:a%=ASC(MID$(A$,i%,1)):IF (a%>
64 AND a%<91) a%=a% AND 223:a%=a%+159
 15ØØ MOVEgx%,gy%:PRINTCHR$a%:gx%=gx%+2Ø
-4*(a%=236 OR a%=246):NEXT:VDU4:ENDPROC
 151Ø :
 152Ø DEFPROCpfi(a%,b%,c%,d%)
 153Ø FORj%=a%TOb%:u%=-6*(j%-1)*(j%<6)-6
*(j%-6)*(j%>5):PRINTTAB(u%,c%)fcon$:PROC
small(F$(j%),32*(u%+4),d%):NEXT:ENDPROC
 154Ø :
 155Ø DEFPROCfb(X%,Y%,Z%):X%=X%*32:Y%=1Ø
23-32*Y%:VDU5:GCOL3,1:MOVEX%,Y%:PRINTSTR
ING$(Z%,CHR$255):GCOLØ,1:VDU4:ENDPROC
 156Ø :
 157Ø DEFFNsub:S%=Ø:RESTORE (5ØØØ+M%*1Ø)
:READ A%,B%,C%,D%:PROCop(A%,B%,C%,D%,1):
READ j%:FORi%=1TOj%:READi$:PRINTi$:NEXT:
READj%:RESTORE (6ØØØ+1Ø*M%):T%=T%+5Ø:CAL
L pointer:T%=T%-5Ø:PROCsel(1,j%):IF S%>Ø
 PROCfb(1%,t%,r%-A%+1):PROCd(2Ø)
 158Ø PROCcl:=S%
 159Ø :
 16ØØ DEFPROCd(d%):tm%=TIME:REPEATUNTILT
IME>tm%+d%:ENDPROC
 161Ø :
 162Ø DEFPROCsel(K%,N%):s%=Ø:i%=Ø:IF K%=
1 READ 1%,t%
 163Ø REPEAT:i%=i%+1:IF K%=1 READ r%:s%=
s%+1:t%=t%+1:b%=t%
 164Ø IF K%=2 READ 1%,r%,t%,b%,s%
 165Ø IF (?&7Ø>=1% AND ?&7Ø<=r%) AND (?&
71>=t% AND ?&71<=b%) THEN S%=s%
 166Ø UNTILS%>Ø OR i%=N%:ENDPROC
 167Ø :
 168Ø DEFPROCps:PROCop(Ø,8,25,1,2):PRINT
"POINTER SPEED"'"3Ø (fast) - 15Ø (slow)"
''"Current = ";T%:CALL curon:INPUT"New :
 " tm%:CALL curoff:IF tm%>29 AND tm%<151
 T%=tm%
```



```
Write entry
Sort entries
Delete page
                                    Tel
Nat Eas                          0227 43040
Nationa                          0224-63756
Needham                          0425-488424
Neil's Furnishers                0582-504439
Nelson Mike                      95 4302
Newtown Council                  0243-62333
North Herts Lesiure              98 64377
Northwich Union                  0344 6718

ENGLAND  open
```

```
 1690PROCcl:ENDPROC
 1700 :
 1710 DEFPROCcol(dc%):M%=6:IF FNsub>Ø:IF
 S%-1<>?(&370-dc%) VDU19,dc%,S%-1;Ø;
 1720 S%=Ø:ENDPROC
 1730 :
 1740 DEFPROCsp:PROCptr:FORl%=1TO24:N$(l
%)=FNget(22):T$(l%)=FNget(11):NEXT:PROCd
is:ENDPROC
 1750 :
 1760 DEFPROCdis:FORl%=1TO24:PRINTTAB(1,
3+l%)N$(l%):PRINTTAB(24,3+l%)T$(l%):NEXT
:ENDPROC
 1770 :
 1780 DEFPROCptr:PTR#H%=(P%-1)*792+1:END
PROC
 1790 :
 1800 DEFFNget(n%):a$="":FORi%=1TOn%:a%=
BGET#H%:IF a%<32 OR a%>125 a%=32
 1810 a$=a$+CHR$a%:NEXT:=a$
 1820 :
 1830 DEFPROCwf:PROCptr:FORl%=1TO24:a$=N
$(l%):PROCput(22):a$=T$(l%):PROCput(11):
NEXT:ENDPROC
 1840 :
 1850 DEFPROCput(n%):FORi%=1TOn%:a%=ASC(
MID$(a$,i%,1)):BPUT#H%,a%:NEXT:ENDPROC
 1860 :
 1870 DEFFNconf(qx%,d$,c$):S%=Ø
 1880 PROCop(qx%,6,qx%+22,1,2):PRINTd$:P
RINT"Please confirm:":PROCqb(qx%+2,5,c$)
:PROCqb(qx%+10,5,"Cancel"):REPEAT
 1890 CALL pointer:IF (?&70>=qx%+2 AND ?
&70<=qx%+7) AND ?&71=5 S%=1:qx%=qx%+2
 1900 IF (?&70>=qx%+10 AND ?&70<=qx%+15)
 AND ?&71=5 THEN S%=2:qx%=qx%+10
 1910 UNTILS%>Ø:PROCfb(qx%,5,6):PROCd(3Ø
):=S%
 1920 :
 1930 DEFPROCdp:d%=Ø:IF FNconf(8,"DELETE
 PAGE","delete")=1:FORl%=1TO24:N$(l%)=S$
:T$(l%)=STRING$(11," "):NEXT:PROCwf:d%=1
 1940 S%=Ø:PROCcl:IF d%=1 PROCdis
 1950 ENDPROC
 1960 :
 1970 DEFFNic:S%=Ø:CALL pointer:RESTORE1
990:PROCsel(2,nf%):IF S%>Ø PROCfb(1%,t%,
2):PROCfb(1%,t%+1,2):PROCd(3Ø)
 1980 =S%
 1990 DATA4,5,2,4,1,1Ø,11,2,4,2,16,17,2,
4,3,22,23,2,4,4,28,29,2,4,5,4,5,6,8,6,1Ø
,11,6,8,7,16,17,6,8,8,22,23,6,8,9,28,29,
6,8,1Ø
 2000 :
 2010 DEFPROCcf:PROCop(A%,B%,23,D%,2):PR
INT"CREATE FILE"'"Filename:":N%=7:fn$=FN
inp:IF fn$="" PROCcl:S%=Ø:ENDPROC
 2020 CLOSE#Ø:IF H%>Ø PROCfb(36,V%,3)
 2030 H%=OPENUP fn$:IF H%>Ø VDU7,12:PRIN
T'"FILE EXISTS !":CLOSE#H%:PROCd(1ØØ):PR
OCcl:S%=Ø:H%=Ø:ENDPROC
 2040 CLS:PRINT"Creating file."'"Please
wait...":$str="*SAVE "+fn$+" ØØØØ +52ØØ"
:CALL oscli:PRINT"File created.":CF$=fn$
 2050 H%=OPENUP fn$:PROCd(8Ø):P%=1:PROCc
l:PROCfb(36,2,3):V%=2:PROCs(1):PROCsp:S%
=Ø:ENDPROC
 2060 :
 2070 DEFPROCaz:IF V%>1 PROCfb(36,V%,3)
 2080 V%=?&71:PROCfb(36,V%,3):P%=V%-1:PR
OCsp:ENDPROC
 2090 :
 2100 DEFPROCwrite:q%=Ø:PROCop(23,3Ø,34,
28,3):PROCqb(24,29,"Quit write")
 2110 REPEAT:REPEAT:RESTORE215Ø:CALL poi
nter:PROCsel(2,3):UNTILS%>Ø:l%=?&71+3*(S
%<3):IF S%=1:VDU31,1,?&71:N$(l%)=FNip(22
):VDU31,24,?&71:T$(l%)=FNip(11)
 2120 IF S%=2:VDU31,24,?&71:T$(l%)=FNip(
11)
 2130 IF S%=3 q%=1:PROCfb(24,29,1Ø):S%=Ø
 2140 UNTIL q%=1:PROCwf:PROCcl:ENDPROC
 2150 DATA1,22,4,27,1,24,34,4,27,2,24,33
,29,29,3
 2160 :
 2170 DEFPROCdf:PROCfl:IF nf%>Ø f%=FNic
 2180 PROCcl:IF nf%=Ø OR f%=Ø ENDPROC
 2190 IF FNconf(3,"DELETE FILE","delete"
)=2 THEN GOTO221Ø ELSE $str="*DEL."+F$(f
%):IF F$(f%)=FNpad(7,CF$) CLOSE#Ø:H%=Ø
 2200 CALL oscli
 2210 PROCcl:S%=Ø:ENDPROC
 2220 :
 2230 DEFFNip(n%):N%=n%:qx%=1-23*(n%=11)
:PROCfb(qx%,?&71,n%):A$=FNinp:PROCfb(qx%
,?&71,n%):IF A$<>"" THEN VDU31,qx%,?&71:
a$=FNpad(n%,A$):PRINTa$:=a$
 2240 IF n%=11:=T$(l%)
 2250 =N$(l%)
 2260 :
 2270 DEFFNinp:?&D3=255:*FX200,1
```

→22

MARTIAL ARTS

武道

Martial arts feature strongly among the current top ten best selling computer games. Alan Webster, no mean player with the keyboard, has been chancing his arm (and leg).

Title	**: Yie Ar Kung Fu**
Supplier	**: Imagine Software**
Address	**: 6 Central Street, Manchester M2 5NS.**
Phone	**: 061-834-3939**
Price	**: £9.95**

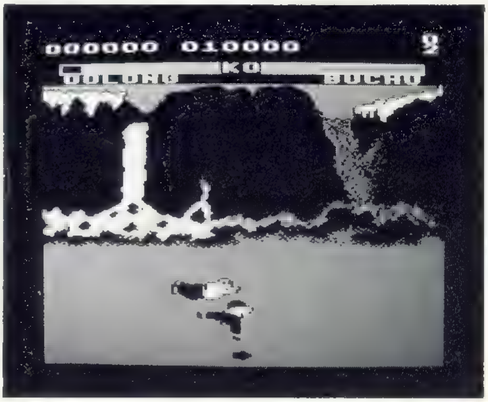
Title	**: Karate Combat**
Supplier	**: Superior Software**
Address	**: Regent House, Skinner Lane, Leeds 7.**
Phone	**: 0532-459453**
Price	**: £8.95 Tape (£11.95 Disc)**

Title	**: Way of the Exploding Fist**
Supplier	**: Melbourne House**
Address	**: 39 Milton Trading Estate, Abingdon, Oxon OX14 4TD.**
Phone	**: 01-943-3911**
Price	**: £9.95**

The BBC software games market contains many sporting simulations, and one of the biggest and most recent successes in this field is the Kung Fu type of game.

There are currently three versions of Kung Fu available for the Beeb, from Imagine, Superior Software and Melbourne House. The object in each game is to defeat a martial arts opponent (or exponent?) by using a series of kicks and punches obtained through several key-presses or movements of the joystick.

A wide variety of moves is possible, including low and high kick, low and high punch, flying kick, back sweep and blocks, plus the usual directional movements. To master all of these can take a very long time. All three games can be controlled from both keyboard or joystick, but I found that Yie Ar Kung Fu was the only one that was easily controlled from the keyboard.



YIE AR KUNG FU

Although Kung Fu is traditionally unarmed combat, this version from Imagine features both armed and unarmed fighting. The object is to beat eight opponents, including the delectable 'Fan', a female who relies heavily on kicking.

To defeat an opponent, you must land eight blows before he or she does. If you manage to defeat your opponent without once being hit yourself, then you score bonus points.

After every fourth opponent, things hot up a little, and numbers of kitchen utensils are hurled at you, including the inevitable wok. To survive, you must either smash or avoid all of these.

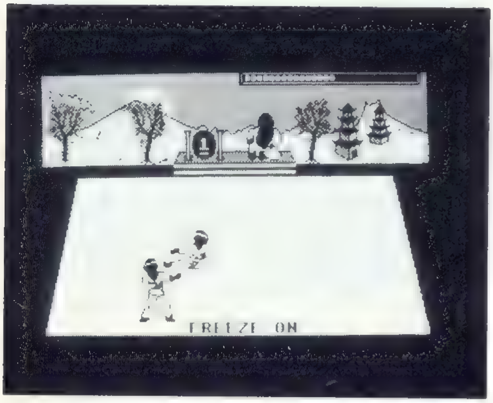
The game is colourful, using an eight colour mode 5 screen, and the action is relatively easy to control from either keyboard or joystick, with only 5 keys needed to launch one of 16 various attacks. My major criticism of this game is that it is perhaps a little too easy to win.

KARATE COMBAT

In Karate Combat, you have twice as many opponents to defeat as in Yie Ar Kung Fu, and as with true Kung Fu, the game concentrates only on hand and foot combat.

As with all three of these packages,

you are allowed the option of keyboard or joysticks, but the seven-key action will likely tie your fingers in knots.



The game offers three modes of play: single player practice, a two-player game, or fighting the computer. And you may also enter a competition with a £100 prize if you get really hot.

Karate Combat makes good use of mode 1 graphics. It plays a challenging game, and represents excellent value for money.

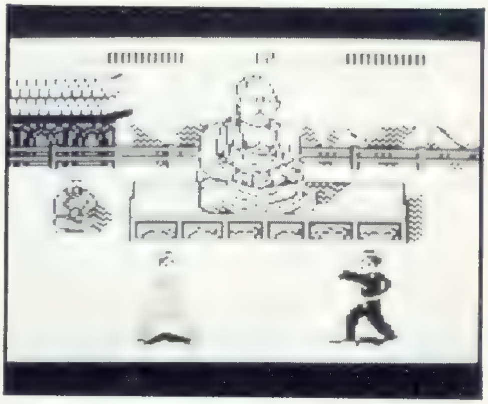
WAY OF THE EXPLODING FIST

This version from Melbourne House has received rave reviews for other machines and looks like doing the same on the BBC.

The opening screens are well conceived, and the viewer is treated to a nicely animated fight between well-trained professionals before he himself is drawn into combat, while the ears are caressed (?) with music of a pleasantly oriental flavour.

Game options are selected with the function keys, and if you elect to play against the computer your opponents become progressively more proficient, advancing from novice to seventh dan.

If you have no joysticks you will be at a serious disadvantage when playing against the computer, and your attempts to simultaneously coordinate the EIGHT keys required will produce apoplexy.



Not withstanding, Way of the Exploding Fist is for my money the best of the three games reviewed here. Its magnificent graphics and animation combined with just the right degree of difficulty give it a clear edge over the opposition.

POINTS ARISING POINTS ARISING POINTS ARISING POINTS

PERSONAL DIARY MANAGER (BEEBUG Vol.4 No.7)

Although not part of the original program, the "Exit to menu" option can be extended to work with options 2,3 and 4 by adding the following lines to the main program:

```
1195 MM%=0                        1755 IF MM% X%=366:MM%=0
1675 IF MM% X%=366:MM%=0          1835 IF MM% X%=366:MM%=0
```

BEEBUG Filer (BEEBUG Vol.4 Nos6 - 8)

A small bug has emerged in the sort routine. Change lines 8980 and 9000 as follows:

```
8980 R=0:REPEAT:R=R+1:UNTIL p$=field$(R) OR R=f
9000 IF R=f AND p$<>field$(R) PRINT"No such field":ENDPROC
```

A similar bug also exists in the FNfield function which can be corrected by changing the following two lines:

```
21620 REPEAT:I=I+1:UNTIL p$=field$(I) OR I=f
21640 IF I=f AND p$<>field$(I) THEN =" " ELSE =record$(I,R)
```

These corrections have been incorporated in Filer on the magazine cassette/disc.

Mail-Merge with Filer

Filer is the popular and powerful database program published in BEEBUG Vol. 4 No. 6 to Vol. 4 No. 8. In this short update, Mike Williams shows how Filer may be modified to provide a useful mail-merge facility for those with any suitable wordprocessor or editor (including View and Wordwise).

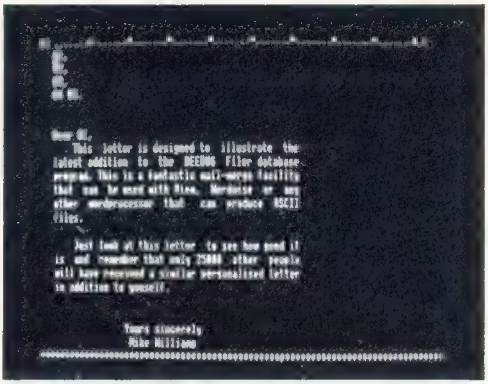
BEEBUG Filer, the database program that was described in the November to Jan/Feb issues of the magazine can be readily extended or modified to provide additional features. Such extras often cost little in code, as frequent use can be made of existing procedures and functions, a big advantage of the modular approach adopted.

This month we will see just how easy it is to add a mail-merge facility for use with View or Wordwise. Indeed, any word processor or editor that can produce ASCII files will do.

To explain what is meant by 'mail-merge' consider the situation where we want to send the same letter to a number of different people whose names and addresses are all held in a datafile. We can use a word processor to create the letter and mark in some way where the name and address of the recipient are to appear. A mail-merge option will then merge the text file and the datafile together to produce as many copies of the letter as required, each individualised with information from the datafile.

This could be simply the name of the addressee, but could also include other data as well. In this particular version up to 9 different fields may be referenced at any one time, and Filer's other features may be used to select which particular records from a given database will be used in any mail-merge operation.

To add the mail-merge facility to Filer, load your existing version of Filer into memory and then add the lines listed at the end of this article (for Basic I



```
Dear @1,
   This letter is designed to illustrate the
latest addition to the BEEBUG Filer database
program. This is a fantastic mail-merge facility
that can be used with View, Wordwise or any
other wordprocessor that can produce ASCII
files.

   Just look at this letter to see how good it
is and remember that only 25000 other people
will have received a similar personalised letter
in addition to yourself.

            Yours sincerely
             Mike Williams
```

Text File

change OPENUP to OPENIN). This produces the updated version of Filer, but it is advisable to save this separately to your original version until you have checked that your new version works correctly.

To use this new facility, create some text with your word processor, marking each point where data is to be inserted from the datafile by @1, @2 etc. Once this has been done, create a spooled version of the text (option 8 in Wordwise, remove rulers and other commands in View). Run Filer and enter the commands:

```
OPEN <datafile>
MERGE <textfile>
```

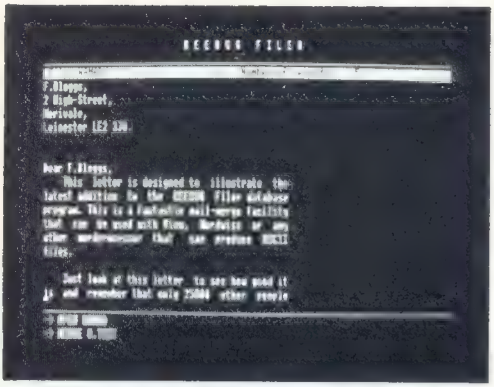
For each record in the datafile, the text will be printed out with @1 replaced by the contents of the first data field, @2 by the second and so on. If you want to change the marker character (@), you can do so with the command:

```
MARK <character>
```

specifying the character of your choice. Up to nine (@1 to @9) different fields can be inserted into any one piece of text, and each data field can occur as many times as required. The SELECT option can also be used as described in previous Filer articles (BEEBUG Vol.4 No.8) in order to merge only selected records with a given text file.

PRACTICAL POINTS

If you avoid all unnecessary spaces when adding this amendment to Filer, and run Filer with PAGE set to &1400 (type: PAGE=&1400 <Return>), you should encounter no memory problems. Note that as the Filer mail-merge facility is implemented quite simply it may upset any previous formatt-



```
BEEBUG FILER

F.Bloggs,
2 High-Street,
Merivale,
Leicester LE2 3JN.

Dear F.Bloggs,
    This letter is designed to illustrate the
latest addition to the BEEBUG Filer database
program. This is a fantastic mail-merge facility
that can be used with View, Wordwise or any
other wordprocessor that can produce ASCII
files.

    Just look at this letter to see how good it
is and remember that only 25000 other people
[illegible]
[illegible]
```

Merged File

ing of the text file. This particularly affects right justification of text. This is virtually unavoidable unless arrangements were made to pass the mail-merged text back through the word processor, or comparable formatting routines were to be included within Filer itself.

When used with View files, Filer has no way of knowing where tab stops were set. All tab characters are therefore ignored and should be replaced in the original text by spaces.

SPOOLING DATA FROM FILER

It may be worth pointing out that Filer already has the ability to transfer data into a file suitable for reading in by a word processor. This can be useful when compiling reports in which it is desirable to include output from a datafile. To do this will require the use of a format file and essentially follows the same pattern as when outputting to a printer or to the screen. To output data to a file called DATA from a datafile called RECORDS, proceed as follows:

```
   OPEN RECORDS
   *FX5,0
   FORMAT FORM1
   *SPOOL DATA
   PRINT
   *SPOOL
```

FORM1 is any suitable format file. The resulting file DATA can then be read in using View or Wordwise.

MAGAZINE CASSETTE/DISC

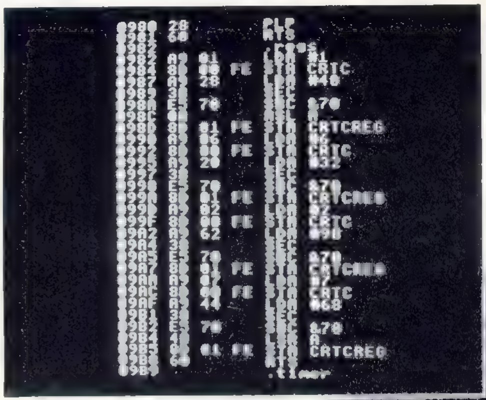
If you do not already have a copy of BEEBUG Filer, then a complete version is included on this month's magazine cassette/disc, and this includes the mail-merge feature described above. The additional 'Filer Notes' are also still available, as mentioned in BEEBUG Vol.4 No.8, on receipt of an A5 SAE. We also hope to publish further updates to the Filer program from time to timè. A program to display the contents of a datafile graphically is already well advanced.

SUMMARY OF BEEBUG FILER COMMANDS

CREATE <filename>	Create a datafile
CREATE <filename>/n	Create a format file
OPEN <filename)	Open a datafile
CLOSE	Close current datafile
EXTEND <filename>	Extend a datafile
ADD	Add a record
DELETE n	Delete record n
UPDATE n	Update record n
DISPLAY	Display all records
DISPLAY n	Display record n
FORMAT <filename>	Select output format
PRINT	Print all records
PRINT n	Print record n
SELECT <search>	Specify search string
SORT <data field>	Sort on data field
MERGE <text file>	Perform mail-merge
COMMANDS	List all commands
END	Exit from Filer
* <command>	Perform any *command

```
 1050 delete%=0:search$="-1":mark=64
 1100 DATA 15
 1170 DATA UPDATE,SORT,SELECT,EXTEND,MER
GE,MARK
 1770 IF C=15 THEN PROCdisplay(pm$,2)
 1780 IF C=16 THEN mark=ASC(pm$)
 4820 LOCAL G,I,start,end,n:IF flag<2 TH
EN n=VAL(p$) ELSE F1=OPENUP(p$):IF F1=0
THEN PRINT"No such text":ENDPROC
 4890 IF flag=1 AND NOT format% THEN PRI
NT"No format":ENDPROC
 4970 IF flag=0 THEN PROCdisplay1(I) ELS
E IF flag=1 THEN PROCprint1(I) ELSE IF f
lag=2 THEN PROCmerge1(I)
 4980 IF I<end AND flag>2 THEN G=GET
 5010 IF flag=2 CLOSE#F1
10800 DEF PROCmerge1(n)
10820 LOCAL char:PTR#F1=0
10840 REPEAT:char=BGET#F1
10860 IF char=mark THEN n=BGET#F1-48:PRI
NT FNstrip(record$(n,0),".");ELSE IF cha
r>31 AND char<128 VDUchar
10870 IF char=13 THEN VDU13,10 ELSE IF c
har=26 VDU32
10880 UNTIL EOF#F1:VDU12:ENDPROC
10900 :
```

SOFT SCREEN SHUFFLE



Many of you will have seen the fancy screen shuffle employed by some commerical games, where an apparently random screen dissolves into a detailed graphics picture. Now you can achieve similar effects with these short routines by Geoff Bains.

Many commercial games perform minor miracles with the Beeb's display. This short program enables you to imitate one of the most popular effects. Micro Power's Ghouls was one of the first with this feature. Each screen of the game was drawn invisibly and then brought into view with the whole screen swirling around. This makes a startling and effective intro that is easily added to your own programs.

The effect is achieved by manipulating the 6845 display chip's registers, namely the vertical and horizontal display and position registers (1, 2, 6, and 7). By decrementing the display registers to zero, the size of the picture is reduced. Decrementing the two position registers to half their normal value as the display registers are changed keeps the picture in the centre of the screen. For further information on the 6845 CRTC see BEEBUG Vol.1 No.8 Page 29. All this can be done in Basic, but an interrupt driven machine code routine allows you to be doing other things while the picture is shrinking or growing.

Type in the program and save a copy in case of any disastrous mistakes. Type RUN <Return> to see the effect. As it stands the program shrinks and grows the mode 1 screen containing the assembler listing. If you incorporate the procedure PROCassem into your own programs and call it once to assemble the code (placed at &0900, though this can be changed by altering line 1060) a mode 0, 1 or 2 screen can be shrunk with CALL shrink and it will be restored back to its normal self with CALL grow.

To save the machine code only, type:
```
*SAVE SHUFFLE 900 +FF <Return>
```
To activate the screen shuffle from your own program, you will need to include some lines at the start to set up 'shrink' and 'grow' and to *LOAD the machine code. You can then shrink and grow your selected screen display at will.

For example, if you have a mode 2 graphics screen *SAVEd as SCREEN, your program could begin:
```
shrink=&900:grow=&919
MODE 2
*LOAD SHUFFLE
CALL shrink
*LOAD SCREEN
CALL grow
```

←17

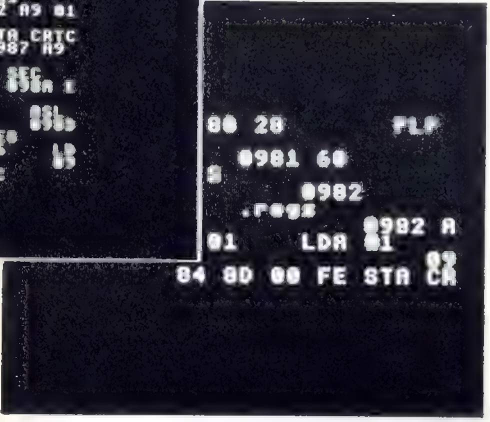
Windows and Icons

```
 2280 CALL input:?&D3=0:*FX200,0
 2290 =$str
 2300 :
 2310 DEFFNpad(L%,a$):IF LENa$=L%:=a$
 2320 L%=L%-LENa$:FORi%=1TOL%:a$=a$+" ":
NEXT:=a$
 2330 :
 2340 DEFPROCsort:REPEAT:e%=0:FORi%=1TO2
3:IF (N$(i%)=S$ AND N$(i%+1)<>S$) PROCex
:GOTO2360
 2350 IF (N$(i%)<>S$ AND N$(i%+1)<>S$):I
F N$(i%)>N$(i%+1) PROCex
 2360 NEXT:PROCdis:UNTILe%=0:IF FNconf(8
,"WRITE TO FILE?","write ")=1 PROCwf
 2370 PROCcl:S%=0:ENDPROC
 2380 DATA9,13,5,5,1,16,21,5,5,2
 2390 :
```

The process can be speeded up or slowed down by altering the four byte value in line 1480. Try &FFFFFFFE for a very fast change around. With this routine you can now give your software that truly professional touch.

```
NJ   10 REM PROGRAM SHUFFLE
   20 REM VERSION B0.2
   30 REM AUTHOR  GEOFF BAINS
   40 REM BEEBUG  MAY 1986
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60
  100 MODE1
  110 PROCassem
  120 FOR I%=6 TO 0 STEP -1
```

```
PLP
0981 60
.regs
0982 A9 01
FE STA CRTC
0987 A9
SEC
CRTCREG
A9 06
CRTC
```



```
80 20        PLP
  0981 60
S
        0982
  .regs
             0982 A
01      LDA  01
84 8D 00 FE STA CR
```

```
 130 I=INKEY(300):CALL shrink
 140 I=INKEY(300):VDU19,0,I%;0;:CALL grow
 150 NEXT I%
 160 END
 170 :
1000 DEF PROCassem
1010 EVNTV=&0220
1020 OSBYTE=&FFF4
1030 OSWORD=&FFF1
1040 CRTC=&FE00
1050 CRTCREG=&FE01
1060 FOR pass=0TO3STEP3:P%=&0900
1070 [OPTpass
.shrink
1090   LDA #in MOD 256:STA EVNTV
1100   LDA #in DIV 256:
       STA EVNTV+1
1110   LDA #14:LDX #5:
       JSR OSBYTE
1120   LDA #0:STA&70:
       JSR set:RTS
1130 .grow
1140   LDA #out MOD 256:STA EVNTV
1150   LDA #out DIV 256:STA EVNTV+1
1160   LDA #14:LDX #5:JSR OSBYTE
1165   LDA #32:STA&70
1170 .set
1180   LDX #timer MOD 256
1190   LDY #timer DIV 256
1200   LDA #4:JSR OSWORD
1210   RTS
1220 .in
1230   PHA:TXA:PHA:TYA:PHA:PHP
1240   JSR regs
1250   INC &70:LDA &70:CMP #33:BNE ret
1260   LDA#1:STA CRTC:LDA#0:STA CRTCREG
1270   LDA #13:LDX #5:JSR OSBYTE
1280   .ret JSR set
1290   PLA:TAY:PLA:TAX:PLA:PLP:RTS
1300 .out
1310   PHA:TXA:PHA:TYA:PHA:PHP
1320   JSR regs
1330   DEC &70:BPL ret2
1340   LDA #13:LDX #5:JSR OSBYTE
1350   .ret2 JSR set
1360   PLA:TAY:PLA:TAX:PLA:PLP:RTS
1370 .regs
1380   LDA #1:STA CRTC
1390   LDA #40:SEC:SBC &70:ASL A:STA CR
TCREG
1400   LDA #6:STA CRTC
1410   LDA #32:SEC:SBC &70:STA CRTCREG
1420   LDA #2:STA CRTC
1430   LDA #98:SEC:SBC &70:STA CRTCREG
1440   LDA #7:STA CRTC
1450   LDA #68:SEC:SBC &70:LSR A:STA CR
TCREG:RTS
1460 .timer
1470 ]:NEXT pass
1480 !P%=&FFFFFFFB
1490 P%?5=&FF
1500 ENDPROC
```

```
 2400 DEFPROCex:N$(0)=N$(i%):T$(0)=T$(i%
):N$(i%)=N$(i%+1):T$(i%)=T$(i%+1):N$(i%+
1)=N$(0):T$(i%+1)=T$(0):e%=e%+1:ENDPROC
 5000 :
 5010 DATA0,8,19,1,5,ADJUST..," ",pointe
r speed,background colour,foreground col
our,3
 5020 DATA3,6,16,1,3,Load file,Delete fi
le,Create file,3
 5030 DATA8,6,22,1,3,Write entry,Sort en
tries,Delete page,3
 5060 DATA0,11,12,1,8,black,red,green,ye
llow,blue,magenta,cyan,white,8
 6000 :
 6010 DATA1,3,13,17,17
 6020 DATA4,1,13,14,14
 6030 DATA9,1,19,20,19
 6060 DATA1,1,5,3,5,6,4,7,4,5
```

DO-IT-YOURSELF BASIC (Part 2)

Alan Webster shows how to exploit the basic program described last month by implementing several new and interesting additions to the Basic language.

Last month we discussed the basic program needed to decode and implement new Basic keywords. This month we shall be adding the detail to that program to show how to implement several new Basic commands and instructions. In each case the additional code should be added to last month's program.

CAPS LOCK AND SHIFT LOCK

The first new commands to be covered provide program control of Caps Lock and Shift Lock, allowing them to be switched on and off. The four new commands are:

```
      CAPSON      CAPSOFF
      SHIFTON     SHIFTOFF
```

These commands will select the relevant keyboard mode, and reflect this in the two L.E.D. lights on the casing.

When one of the new keywords is met, the machine code jumps to the relevant routine to select the relevant keyboard value for the Caps Lock and Shift Lock state. This is then set by using OSBYTE 202. OSBYTE 118 is then used to reflect the Caps Lock and Shift Lock status in the keyboard L.E.D. lights. Here is the additional code needed for this:

```
3040 OPT FNEQUS("CAPSON")
3050 OPT FNEQUB(0)
3060 OPT FNEQUW(capson)
3070 OPT FNEQUS("CAPSOFF")
3080 OPT FNEQUB(0)
3090 OPT FNEQUW(capsoff)
3100 OPT FNEQUS("SHIFTON")
3110 OPT FNEQUB(0)
3120 OPT FNEQUW(shifton)
3130 OPT FNEQUS("SHIFTOFF")
3140 OPT FNEQUB(0)
3150 OPT FNEQUW(shiftoff)
4500 .capson :JSR chkend
4520 LDX #&20
4530 JMP reflect
4550 .shifton
4560 JSR chkend
4570 LDX #&10
4580 JMP reflect
4600 .capsoff
4610 .shiftoff
4620 JSR chkend
4630 LDX #&30
4650 .reflect
4660 LDA #202
4670 LDY #0          \Set Caps/
4680 JSR &FFF4       \Shift-Lock
4690 LDA #118
4700 JSR &FFF4       \Set LED status
4720 JMP cont
```

[Basic II users may omit the OPT FN part of lines 3040 to 3150. This means that line 3040 would be: EQUS("CAPSON").]

LISTING OS VECTORS

The next keyword to be added is VECTORS. This will print out the values of the operating system vectors in page 2 of memory (useful information for programmers). These vectors can be found on page 254 of the Advanced User Guide.

```
3160 OPT FNEQUS("VECTORS")
3170 OPT FNEQUB(0)
3180 OPT FNEQUW(vectors)
5000 .vectors
5010 JSR chkend
5020 LDY #0          \Vector offset
5030 .veclp
5040 LDA #ASC("(") \Print (
5050 JSR &FFEE
5060 LDA #ASC("&") \Print &
5070 JSR &FFEE
5080 LDA #ASC("2") \Print 2
5090 JSR &FFEE
5100 TYA
5110 JSR phex        \Print vector lo
5120 LDA #ASC(")") \Print )
5130 JSR &FFEE
5140 LDA #32         \Print Space
5150 JSR &FFEE
5160 LDA #ASC("=") \Print =
5170 JSR &FFEE
5180 LDA #32         \Print Space
5190 JSR &FFEE
5200 LDA #ASC("&") \Print &
5210 JSR &FFEE
5220 LDA &200,Y      \Get vector
5230 STA &70         \value lo.
5240 LDA &201,Y      \Get vector
5250 STY &71         \value hi.
5260 JSR phex        \Print hi byte
5270 LDA &70
5280 JSR phex        \Print lo byte
5290 LDY &71
5300 JSR &FFE7       \Newline
```

```
5310 INY:INY
5330 CPY #&36
5340 BNE veclp      \Next vector
5360 JMP cont
```

Add these lines together with those for CAPSON/CAPSOFF etc. to the main program and re-run this. Then type CALL &900 <Return> to set up the machine code and add the new keywords to Basic.

All five new keywords (CAPSON, CAPSOFF, SHIFTON, SHIFTOFF and VECTORS) may now be tested. This is most readily done in immediate mode. Try CAPSON and CAPSOFF, typing in any text from the keyboard to see the effect. To list the vectors just type VECTORS <Return>. You should notice that the BRK vector (&202) is set to our own machine code routine, and not its default value.

The VECTORS routine simply loops from &200 to &234 in steps of two, reading the values of these vectors and printing them in hexadecimal using Basic's 'print accumulator as two byte hex' routine (phex).

ITALIC PRINTING

The last example of a new instruction is the keyword ITALIC to turn italic printing on and off. The keyword takes an extra parameter, which should be 0 to turn italics off and 1 to turn italics on. The keyword on its own will also enable italic printing.

To use this in a program, or in immediate mode, simply type the command, and subsequent output will be in italic or normal printing as selected.

The italic routine works in any graphics mode by re-defining character 255, rotating the top 3 lines of the character left and the bottom 3 lines right. The new character is then printed in place of the old character.

The routine will not print italic characters in mode 7, and only re-defines characters 32 to 254 with the exception of the delete character (127).

As an example, add this further code to the main program and re-run once again before typing CALL &900. Type 'ITALIC 1' in immediate mode and select mode 6. Then type in any text, and this should appear in italics. Typing 'ITALIC 0' should return the machine to its normal text state.

As the routine re-directs the 'WRite CHaracter Vector' (WRCHV), not only is keyboard input displayed in italics, but also any other text that is output (e.g. LISTing a Basic program).

This example, though only a simple one, shows that it is also possible to pass parameters with a new keyword. In this case, the parameter has to be either 0 or 1 (with or without leading spaces). More complex parameter decoding would take up much more code.

The relevant code for the new ITALIC keyword is as follows:

```
3190 OPT FNEQUS("ITALIC")
3200 OPT FNEQUB(0)
3210 OPT FNEQUW(italic)
6000 .italic
6010 LDY &A          \Statement offset
6020 .itlp
6030 LDA (&B),Y      \Get next char.
6040 INY
6050 CMP #32         \Spaces not needed
6060 BEQ itlp
6070 CMP #13         \Ret., use ITALIC1
6080 BEQ italok2
6090 CMP #49         \=1, use ITALIC1
6100 BEQ italok
6110 CMP #48         \=0, use ITALIC0
6120 BNE italn
6130 JMP italic0
6150 .italn
6160 BRK             \Not blank 0 or 1
6170 OPT FNEQUB(16)
6180 OPT FNEQUS("Syntax error")
6190 BRK
6210 .italok2
6220 DEY             \Adjust PTR#A
6230 .italok
6240 STY &A          \Reset PTR#A
6250 JSR chkend
6270 LDX #indir MOD 256    \Italic
6280 LDY #indir DIV 256    \routine
6290 CPX &20E        \Has italic been
6300 BNE setit       \set up?
6310 CPY &20F
6320 BEQ setup
6330 .setit
6340 STX &20E        \Set OSWRCH vector
6350 STY &20F
6370 .setup
6380 JMP cont
```

```
6400 .indir
6410 STA chrbuf      \Store char
6420 PHA:TYA:PHA     \Preserve A,X,Y
6450 TXA:PHA
6470 LDA &355        \Test for mode 7
6480 CMP #7
6490 BEQ noit
6500 LDA chrbuf      \Test for control
6510 CMP #32         \character (<32)
6520 BCC noit
6530 CMP #127        \Test for delete
6540 BEQ noit
6550 JMP ysit
6560 .noit
6570 PLA:TAX:PLA     \Restore A,X,Y
6600 TAY:PLA         \and exit to
6620 JMP osbyte      \default OSWRCH.
6630 .ysit
6640 LDX #chrbuf MOD 256
6650 LDY #chrbuf DIV 256
6660 LDA #10         \OSWORD 10, read
6670 JSR &FFF1       \char definition
6680 LSR chrbuf+1    \Shift top 3
6690 LSR chrbuf+2    \lines of char
6700 LSR chrbuf+3    \left, and
6710 ASL chrbuf+6    \bottom 3 lines
6720 ASL chrbuf+7    \right.
6730 ASL chrbuf+8
6740 LDA #23         \Redefine char
6750 JSR osbyte
6760 LDA #255        \255 to be our
6770 JSR osbyte      \italic char.
6780 LDY #1
6790 .chrlp
6800 LDA chrbuf,Y
6810 JSR osbyte
6820 INY
6830 CPY #9
6840 BNE chrlp
6850 LDA #255        \Output italic
6860 JSR osbyte      \character
6880 PLA:TAX:PLA     \Restore A,X,Y
6910 TAY:PLA:RTS     \Return.
6950 .vect2
6960 OPT FNEQUW(!&20E)
6970 .chrbuf
6980 OPT FNEQUS("000000000")
7000 .italic0
7020 STY &A          \Reset PTR#A
7030 JSR chkend
7050 LDA vect2       Reset OSWRCH
7060 STA &20E        vector, to turn
7070 LDA vect2+1     \off italics
7080 STA &20F
7100 JMP cont        \Exit.
9050 osbyte= &E0A4
9550 osbyte= &E0A4
```

SAVING THE CODE

To save the machine code generated by the complete program (i.e. last month's listing plus any or all of the three additions listed here), you will first need to find the start and end addresses of the code. These are displayed when the program is run. If for example, the two addresses are &900 and &B57 for start and end respectively, then you will need to type:

```
*SAVE BASX 900 B57 <Return>
```

to save the machine code in the file BASX.

When you wish to use these Basic extensions in a program of your own, simply include the line *RUN BASX at the start.

At the moment, the machine code for these extensions occupies the function key buffer at &B00 (Econet workspace on the Master 128). To alter the address at which the code is assembled, change the value of P% in line 1070 of the original program.

Now you have the basic framework for adding any of your own commands to Basic. Indeed, if you have any particularly good ideas why not drop us a line and let us know.

BASIC I

Basic I users will need to add the three functions listed below to both last month's program and this month's extended version to overcome the lack of EQUB, EQUS and EQUW in the Basic I assembler. The lines to do this are as follows and these should be appended to the program.

```
10000 DEF FNEQUB(B%)
10010 ?P%=B%:P%=P%+1
10030 =PASS
10050 DEF FNEQUS(S$)
10060 $P%=S$:P%=P%+LEN(S$)
10070 =PASS
10090 DEF FNEQUW(W%)
10100 !P%=W%:P%=P%+2
10110 =PASS
```

You will also need to modify five of the original lines as follows:

```
3010 OPT FNEQUS("VARS")
3020 OPT FNEQUB(0)
3030 OPT FNEQUW(vars)
3500 OPT FNEQUB(255)
3530 OPT FNEQUW(!&202)
```

Printer Survey

This month, BEEBUG brings you upto date on the latest printers that have appeared for the Beeb since our last survey in BEEBUG Vol. 3 No. 8. Simon Williams reports on his findings.

In the period since our last printer survey (Vol.3 No.8), several new machines have been introduced. Two trends are noticeable among dot-matrix printers: an increasing dependence on the industry standard set by Epson, including Near Letter Quality, and the decreasing cost of fully-featured printers. We've chosen five machines which reflect these trends.

In addition, we've included two competitively priced daisywheel printers, which offer high quality print for letters and other correspondence.

<u>SEIKOSHA SP-1000A</u>

This is a low-cost Epson compatible printer, offering many of the features of its more expensive companions, and only sacrificing speed and a certain feeling of robustness.

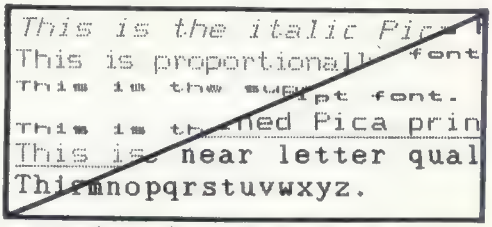
The SP-1000A has the smallest dimensions of any of the printers in the survey, but still manages to take paper up to 10 inches wide. The printer will accept single sheets, which it automatically feeds through to a top of form position when you move the bail bar forward. It also comes as standard with a clip-on tractor unit for continuous paper.

The printer supports all the character sets of the Epson FX80 series, including proportional spacing and down-loadable characters. In addition, it will print Near Letter Quality text, selected either from the printer's control panel or through software. The NLQ typeface is very similar to, though slightly smaller than, Epson's own.

The printer is very easy to use and has a good 'feel' to it. It is, however, a very lightweight mechanism, and I wonder what the print quality would be like after a year or so's use.

The manual supplied runs to 90 pages and covers the functions of the printer concisely, without much of the 'Japlish' which plagued earlier Seikosha manuals.

The SP-1000A is quite slow by modern dot-matrix standards, at only 100 characters per second in draft mode. It is, however, the fastest graphics printer tested, completing the sample dump in 320 seconds. Overall, very good value.


This is the italic Pic
This is proportionall font
This is the superscript font.
This is th ned Pica prin
This is near letter qual
Thi mnopqrstuvwxyz.

Seikosha/Amstrad Print Samples

<u>AMSTRAD DMP2000</u>

This is a printer that BBC Micro owners might not immediately think of. Although produced for use with Amstrad's own micros, it is fully Epson compatible, and uses a standard Centronics interface. You will need a ribbon lead with a BBC Micro plug on one end, but otherwise the printer can be used like any other.

The printer is housed in a dark grey plastic case and is not the most attractive peripheral ever made. It has a clear Perspex cover reminiscent of Mr Sugar's cheaper turntables and is shaped more like a flat-bed plotter than a printer. This is not accidental, as its print head is mounted above the paper and prints down onto it. This technique has several advantages. It allows single sheets to be fed in easily, and text can be printed on thin card, as it doesn't have to wrap around a print roller. Fanfold paper can also be fed in using the built-in tractors, and stored under the printer, which can be raised up quite securely on two folding legs. The print head itself is very light, and similar to that in the SP-1000A.

The DMP2000 supports Epson control codes, and can print in all the standard modes including NLQ. The typeface is not quite as attractive as on the Seikosha, but graphics dumps are as fast.

The manual supplied is thick and well illustrated. It covers connecting and setting up the printer with the BBC and a number of other micros, and gives full details of the commands necessary to operate it.

The DMP2000 is a very good printer for the low asking price. Although a little slow in operation, it has several redeeming features, in particular its paper handling and graphics speed.

EPSON LX-80

This is Epson's answer to the flood of cheap Epson-compatible printers. It mimics the features of the FX-80, and has NLQ print built-in. It looks rather more stylish than earlier Epsons, but is still quite deep.

As supplied it can only take cut sheet paper, as the tractor unit, which fits on top, costs an extra £20. This is very surprising, and seems an odd way to economise. There can be few users of dot-matrix printers who print only on single sheets.

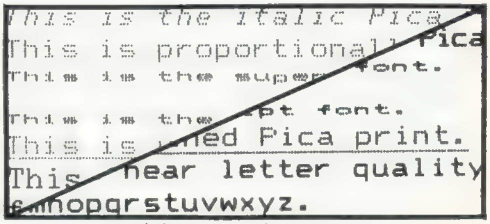
The LX-80 can produce all the normal Epson character sets, with the exception of proportional spacing. Maybe few people use this feature, but it still seems peculiar to leave it out, when it's just machine-code in a ROM.

Several of the more common typestyles can be selected from the front panel by pressing combinations of the FF and LF buttons. This allows selection of a font other than standard Pica, even if the software in use makes no provision for it. The Epson NLQ font is a very readable Roman style typeface made in two passes of the print-head.

The print speed is a conservative 100 cps, and the test printer dump was a full half minute slower than on the Seikosha or Amstrad printers. Against that, though, it did produce by far the most even graphics dump, with very little 'white lining' between each pass of the print head.

Epson's manuals have always been good, and the one for the LX-80 is no exception. It's a smaller format than the others, but is spiral bound with over 170 pages.

A couple of annoying niggles with tractors and proportional spacing, but if you want an Epson printer...


This is the italic Pica
This is proportional Pica
This is the super font.
This is the pt font.
This is ned Pica print.
This near letter quality
nopqrstuvwxyz.
Epson/Micro-P Print Samples

MICRO PERIPHERALS MP-165

This is Micro Peripherals' own-brand machine, based loosely on several other high quality Japanese designs. It's quite tall at 5 inches, although it has a modest 'footprint' and is built like a tank. This is no bad thing for a printer, and the even print sample reflects the precision of its construction.

The control panel, on the left of the platen at the front, contains the normal line feed, form feed and on-line selection buttons. In addition, you can select the NLQ font from here, or by sending Epson Escape sequences.

The printer is fully Epson compatible in its other features, including loadable characters and proportional spacing.

The tractors for pin-feed paper are positioned behind the roller, so paper is effectively pushed through the mechanism. Single sheets can also be fed in, although this is quite awkward. The mechanism on the review machine was stiff and difficult to move by hand.

The printer is rated at 165 cps and it certainly moves quickly when printing in draft mode. The NLQ font is not Roman like most of the others, but is a high quality version of the standard Pica typeface. It is the nearest to daisywheel quality of any of the dot-matrix printers tested.

The manual is exemplary. Each Escape sequence is given at least half a page, with print samples and example listings.

The MP-165 is not the cheapest printer available, but is very well built, runs fast, and gives the impression of being able to cope with many years of hard use.

JUKI 5520

Juki are best known in this country for their excellent daisywheel printers, but they have now moved into the dot-matrix market with the 5510, and the 5520, which also prints in colour. The 5510 can be upgraded to the colour version for around £120.

The 5520 is a big printer, although its carriage is no wider than normal. The standard controls and indicators are available, although NLQ can only be selected through software.

Some of the space inside the case is to take the colour print mechanism. This consists of two pivoted arms, which move the four-colour ribbon up and down, so that the print head strikes the appropriate coloured band. Overprinting allows up to 8 colours to be printed. No colour print dump routine was supplied, but the diagnostic test produces good coloured text.

The printer is both Epson and IBM compatible, so if you have an IBM PC as well as a BBC (lucky you!), you can use the 5520 directly with each.

The Juki supports all the Epson codes and can handle down-loaded characters. Its own NLQ font is produced in three passes of the print head. The quality of the characters it produces doesn't appear noticeably better than the others, though. The graphics dump is fair, though not particularly quick.

The manual is very well produced; over 150 spiral bound pages including some example coloured dumps. Quite a lot of this is wasted on the BBC Micro user, however, since it includes separate chapters on installation with a number of different machines, mostly American.

If you want a very versatile machine, offering good compatibility and the option of printing in colour, The Juki 5510 offers good value. If you can afford the extra for the 5520, it's cheaper than buying the 5510 and the colour upgrade.

SAMPLE DAISYSTEP 2000

This is the cheapest true daisywheel printer on the market, so you might expect it to miss out on some of the features of more expensive machines of its type. In fact, you only really lose out on speed. The Daisystep has a quoted speed of 18cps, so if you're used to a dot-matrix printer (or more expensive daisywheel) you'll find the Daisystep very slow.

A daisywheel printer has no chance of producing graphics dumps, of course, and the only way you can change the typeface is to fit a new printwheel. While this is quite a simple process on the Daisystep, as the print head swivels up to allow easy access to the wheel, it's not that easy to swap wheels within a document.

The Daisystep is a big machine, mainly due to the 13 inch carriage it uses. This means you can easily print A4 sheets sideways (or A3 sheets). Only friction feed is provided as standard, but both a tractor feeder and automatic sheet feeder are available as extras (not cheap).

Controls are set along the front panel, and include the standard switches, plus a test mode which prints out the character set repeatedly.

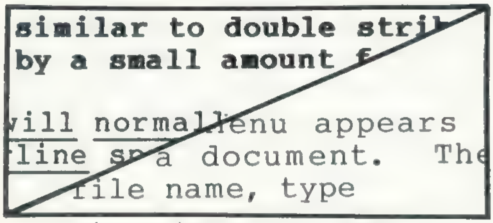
The only typeface options are double strike, bold and underlined print. These are selected by Escape sequence, and the sequences are compatible with the Qume Sprint printer. The Daisystep's ribbons and daisywheels are also interchangable with Qume's.

The manual is a 22 page flimsy pamphlet, readable but with a few grammatical oddities. It describes setting up and using the printer, and includes an interesting ASCII chart, where all the codes are offset by one!

I've been using a Daisystep for over a year now, and find it a solid (if slow) printer, ideally suited for letters to bank managers and magazine editors.

BROTHER HR-15

There are several features which make the HR-15 a top quality daisywheel printer. For a start, its operation is very smooth, and although the hammering of the daisywheel petals is not much quieter than the Daisystep, its carriage feed and head movement is very silky.


similar to double stri
by a small amount f

will normallenu appears
line spa document. The
file name, type

Daisystep/Brother Print Samples

The printer has a 15" carriage to take wide paper and prints at around 18 cps. It will automatically feed single sheets to the top-of-form position, even moving the bail bar out of the way to avoid crushing the paper.

Controls on the front panel include pitch and line spacing (a great advantage over dip switches hidden under a back panel) and a COPY button. This will redump the printer's 3K buffer, sufficient for an A4 page, as many times as you press it - very handy. Line feed, form feed and on-line switches are also present.

Daisywheels and print cartridges are compatible with Diablo printers, as are the control sequences. In addition to the normal black ribbon cartridge, the HR-15 takes a secondary ribbon of a different colour, which can be used to highlight sections of text under software control.

The sample printout from this printer is really superb, well up to anything you could produce on an electronic typewriter. Bold-face and double-strike print is particularly impressive, although there are irregular gaps between the underline characters on single strike underlining. The printer can also space text proportionally.

The manual is clear, concise and covers all the features of the printer adquately. If you want a true correspondence quality printer at a very reasonable price (for a daisywheel), you will probably not find a better deal than the HR-15.

PRINTER SURVEY	SP1000A	DMP2000	LX80	MP165	5520	DS2000	HR15
Print quality	4	3	4	4	4	5	5
Print speed	3	3	2	5	5	1	1
Graphics quality	3	4	5	4	4	-	-
Graphics speed	5	5	2	4	3	-	-
Graphics linearity	1	4	5	3	2	-	-
Feed paper	5	4	3	2	4	4	5
NLQ print	3	3	4	5	4	LQ	LQ
Underline	3	3	3	3	4	5	5
Noise (1= v.noisy)	4	4	3	3	3	2	2
Italics	2	4	3	2	2	-	-
Condensed	2	3	4	4	3	-	-
Elite type	2	3	4	4	3	-	-
Double-width	3	2	3	4	4	-	-
Emphasized (bold)	3	2	2	4	3	5	5
Double Strike	2	3	3	4	4	5	5
Sub/Superscript	2	4	3	4	2	-	-
Graphics characters	2	0	2	0	4(IBM)	-	-
Manual	4	3	4	5	4	2	4
Definable Chars	Y	Y	Y	Y	Y	-	-
Coloured text	N	N	N	N	Y	-	-
Coloured background	N	N	N	N	Y	-	-
Coloured graphics	N	N	N	N	Y	-	-
Proportional spacing	Y	Y	N	Y	Y	N	Y
Tractor feed standard	Y	Y	N	Y	Y	N	N
Approx price inc VAT	£200	£160	£230	£300	£330	£200	£350

All scores in the table above are in the range 0 to 5.

CONCLUSION

Each of the printers reviewed has something to offer. The Seikosha and Amstrad machines are very competitively priced while still offering good compatibility, the LX80 has the Epson name behind it, the MP165 is fast and very robust, and the Juki offers the advantage of colour print, and both Epson and IBM character sets.

Both daisywheels are well-made machines, the Daisystep being considerably cheaper, and the HR-15 offering its useful Copy buffer and auto sheet-feed.

*SPOOL GRAPHICS

The *SPOOL function is one that is often underrated and under-used. Richard Lambley shows how it can provide fast and efficient methods of displaying graphics screens, and save memory into the bargain.

Many BBC Basic programs - not only games - include a lengthy sequence of graphics commands to set up the screen. Often, these can take quite a long time to finish plotting, especially if complicated mathematical calculations are involved.

To speed things up, it's possible to convert these graphics commands into their constituent VDU codes which can then be sent directly to the screen.

A simple way to do this is with the *SPOOL command. Insert in the original program the line *SPOOL <filename> at the start of the graphics sequence, then *SPOOL alone at the end of it.

Run the program and you will find that you have saved as VDU codes all Plot, Draw, VDU, Print or Mode statements between the Spool commands (subject to the usual proviso about Mode and Himem). All other instructions encountered are executed but produce no output to the spool file. You can reload them rapidly from disc using just *TYPE <filename>, making at the same time a useful saving in the memory space that would otherwise be needed for storing them.

The accompanying short program converts this spooled file further into a sequence of EQUD statements which can be *EXECed into an assembler listing (for Basic II or higher). Line 300 marks off the statements into blocks of 256 bytes, ready for OSWRCHing to the screen.

This makes it possible to write graphics routines in Basic and turn them into machine-code quickly and easily.

```
   10 REM PROGRAM *SPOOL GRAPHICS
   20 REM VERSION B0.2
   30 REM AUTHOR  R.LAMBLEY
   40 REM BEEBUG  MAY 1986
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 MODE7
  110 ONERROR CLOSE#0:REPORT:END
  120 INPUT"Source filename",source$
  130 INPUT"Destination filename",dest$
  140 PRINT"Line numbers (Y/N)?";
  150 numbr%=(GET AND &DF)
  160 PRINTCHR$15
  170 IF numbr%=ASC("Y") THENINPUT"First
 line number",numbr% ELSE numbr%=0
  180 chan=OPENIN(source$)
  190 chano=OPENOUT(dest$)
  200 line%=0
  210 REPEAT
  220 I%=0:I$=""
  230 REPEAT
  240 C%=BGET#chan
  250 C$="00"+STR$~C%
  260 I$=RIGHT$(C$,2)+I$
  270 I%=I%+1
  280 UNTILI%=4 OR EOF#chan
  290 I$=" EQUD &"+I$+CHR$&D
  300 IFline% MOD 640=0 THEN I$=".table"
+STR$(line% DIV 640)+I$
  310 IF numbr% I$=FNlines(line%)+I$
  320 PROCsave(I$)
  330 line%=line%+10
  340 UNTILEOF#chan
  350 CLOSE#0
  360 END
  370 :
 1000 DEF FNlines(L%)
 1010 IF numbr% THEN =STR$(L%+numbr%) EL
SE=""
 1020 DEF PROCsave(text$)
 1030 PRINTtext$
 1040 FORJ%=1 TO LEN(text$)
 1050 BPUT#chano,ASC(MID$(text$,J%,1))
 1060 NEXT
 1070 ENDPROC
```

To allow this program to work with Basic I, line 290 should be altered as follows:

```
 290 I$="OPT FNequd(&"+I$+")"+CHR$&D
```

and a further procedure added to your assembler program as follows:

```
10000 DEF FNequd(e%)
10010 !P%=e%:P%=P%+4:=0
```

BEEBUG WORKSHOP - STACKS AND QUEUES

Some issues back we published two articles on the important subject of data structures. Surac adds some more useful techniques to this theme by looking at stacks and queues.

Two articles in BEEBUG Vol.4 Nos.4 and 5 described a range of data structures, such as linked lists and trees, which could be implemented on the Beeb. Such new structures are most useful in simplifying jobs which Basic's standard facilities (variables and arrays) make harder than need be. This month, I will look at a two other simple structures - stacks and queues.

STACKS

Perhaps the simplest new data structure is the 'Last In, First Out' (LIFO) stack. Assembly programmers will have come across this idea already. A stack is an area of memory in which data can be saved, and later recovered. The key factors are that data is stored temporarily and that it comes back in the opposite order to that in which it was saved - hence LIFO.

We can easily set up a stack in Basic, with an array to hold the data and a pointer to show the last item/next space. We have to be able to initialize the stack, to store and to recover data (PUSH and PULL it). We must also avoid trying to save more data than the stack can hold, and trying to pull data from an empty stack. Let's look at some code to do this.

The procedure and functions assume that you have already reserved some memory for the stack using:

```
DIM stack(size)
```

where 'size' is at least as large as the maximum number of items you want to be able to stack.

```
                 BASIC STACK
10000 DEF PROCStackInit(MaxSize%)
10010 sp%=-1
10020 spmax%=MaxSize%
10030 errorval=-9999
10040 ENDPROC

10200 DEF FNPush(val)
10210 LOCAL ok
10220 ok=(sp%<spmax%)
10230 IF ok THEN sp%=sp%+1:
      stack(sp%)=val
10240 =ok

10400 DEF FNPull
10410 LOCAL ok,val
10420 ok=(sp%>=0)
10430 IF ok THEN val=stack(sp%):
      sp%=sp%-1 ELSE val=errorval
10440 =val
```

First, create the stack with PROCStackInit(n), where 'n' is the stack's capacity (not larger than 'size'). The stack pointer ('sp%') is set to the start of the stack and 'errorval' is set up for later use.

To save 'item' on the stack, use:

```
dummy=FNPush(item)
```

FNPush checks that there is room on the stack and, if so, increments the stack pointer and saves 'item'. The routine returns TRUE (-1) if 'item' was saved and FALSE (0) if not (if the stack was full). This gives a way of checking if 'item' was actually saved and maybe taking corrective action.

Pull a value from the stack with:

```
value=FNPull
```

If there is anything on the top of the stack, it is transferred to 'value' and the pointer is reduced by one. If the stack is empty, then the value of 'errorval' is returned. This is -9999, but it could be anything you choose for your own programs. Note that the data still stays in the stack array until eventually overwritten by a PUSH. Only the pointer changes.

That's all there is to a stack, but where would you use one? A good example would be to reverse a stream of input numbers, where the reversing action of the stack's LIFO behaviour makes life easy. How about temporary storage of a series of interim values in a nested or recursive set of PROCedures? Understanding a stack in Basic can also help when it comes to using the same idea in machine code.

QUEUES

We can go on from the idea of a stack to a queue, which is a 'First In, First Out' (FIFO) data structure. It's probably more useful than a stack. It gives a way of temporarily storing data, in the order in which it was received, for later processing.

We can make a queue with an array and two pointers, one showing where the next item into the queue goes, and the other showing the next one to be removed. The trouble there is that, unless you have a HUGE array, you might run off the end of it. A more practical approach is a circular queue. This time, as each pointer goes off the 'top', it is automatically set back to the bottom.

A ring like this has one disadvantage - if there are 'n' places in the queue, you can only recover the last 'n' items to be saved. Earlier data is lost as the pointers cycle round. You must make sure that the array can hold as many items as you are interested in. Another problem is detecting whether the queue is full or empty - if the two pointers indicate the same location, it could be either. Have a think about it and then look at the code.

You can see the similarity to the stack routines. PROCQInit(n) sets up the queue size, initializes the pointers, and sets the flags which the later routines will need.

To add an item to the queue, use:

```
  PROCQ(value)
```

This will always save an item. Since the queue is circular, the earlier stack problem of not being able to add data to a full structure does not apply. Older data is simply overwritten. Line 11210 handles the cycling of 'qwrtptr%', which is where data is added, while line 11230 detects when the queue is full. In the latter case, line 11240 'bumps' the data withdrawal pointer, 'qreadptr%', forever losing whatever it was pointing to.

```
                  BASIC QUEUES
11000 DEF PROCQInit(MaxSize%)
11010 qreadptr%=0
11020 qwrtptr%=-1
11030 qmax%=MaxSize%+1
11040 qerrorval=-9999
11050 qfull=FALSE
11060 qempty=TRUE
11070 ENDPROC

11200 DEF PROCQ(val)
11210 qwrtptr%=(qwrtptr%+1) MOD qmax%
11220 q(qwrtptr%)=val
11230 qfull=((qreadptr%=qwrtptr%) AND
      NOT qempty)
11240 IF qfull THEN
      qreadptr%=(qreadptr%+1) MOD qmax%
11250 qempty=FALSE
11260 ENDPROC

11400 DEF FNUnQ
11410 LOCAL val
11420 IF NOT qempty THEN
      val=q(qreadptr%):
      qreadptr%=(qreadptr%+1) MOD qmax%
      ELSE val=qerrorval
11440 qfull=FALSE
11450 qempty=((qreadptr%-qwrtptr%)=1)
      OR (qwrtptr%=qmax%-1 AND
      qreadptr%=0)
11460 =val
```

To remove data from the ring, use:

```
  value=FNUnQ
```

This gets the next item from the queue and advances qreadptr%, putting it back to the start if necessary (line 11420). If there was no data, the previously set 'qerrorval' is returned.

Where would you use a queue? Anywhere you want to save an initially unknown number of items and later process the last 'n' of them. A good example might be if the Beeb is reading data from the analogue port too fast to deal with at the time. Save it in a queue, and deal with it later. Again, this is an idea that might well be implemented in machine code, but an initial understanding in Basic can make a useful starting point. The magazine cassette/disc has a full demo program of the stack and queue procedures in action.

SIDEWAYS RAM MODULES

Sideways RAM has now become almost de rigeur for the aspiring BBC micro owner who wants to keep up-to-date. Geoff Bains has been looking at several of the cheap sideways RAM modules now available on the market.

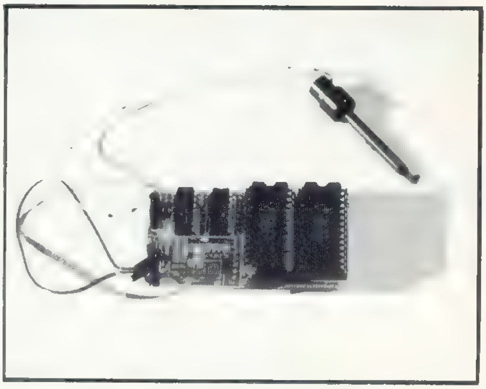
Nowadays, to have a BBC micro with no sideways RAM is almost criminal. As well as allowing you to load up 'ROMs' from disc (even cassette, if you're patient), sideways RAM can be used for a variety of purposes in its own right. You needn't invest in a special RAM board to take advantage of this aspect of the Beeb. There are now many cheap 16K RAM modules available that enable you to stick sideways RAM into any Beeb.

The RAM modules fit into a sideways ROM socket. These were not designed to take RAM and so are not provided with a 'write' line from the processor. This means that a separate 'write' connection must be made. This is simply done with a flying lead with a clip to some other part of the board. No soldering is necessary. You can also fit two or more of these sideways RAM modules in your machine to hold several ROM images at once.

SOFTWARE PIRACY
We are well aware that these devices might be used for illegal copying of software. We neither support nor condone this activity. Software piracy is theft.

PROMICRO
Promicro's sideways RAM module consists of two 8K RAM chips (as used in several ROM/RAM boards such as ATPL's Sidewise) mounted one on top of the other with some clever wiring and a couple of diodes. The whole thing is wrapped in black tape to make it look nice. A flying lead with a clip probe on the end provides the 'write' signal.

Software to load and save ROM images is provided on the instruction sheet along



with clear and comprehensive fitting instructions. The Promicro module is a basic model but none the worse for it.

WATFORD ELECTRONICS
Watford sells two sideways RAM modules: one large and one small. The small model is made from two half sized RAM chips and another minute logic chip on a tiny circuit board the size of a single ROM chip. The half sized chips make the module very small and neat and it will fit into any conceivable situation. A flying 'write' lead has a clip probe on the end. The instructions are in the form of a four page A5 sheet and the software comes on a 40/80 track disc.

The software is menu driven and provides facilities for listing the ROM images present in your machine, saving ROM images, and loading ROM images.

A useful feature is a program to turn the RAM into a printer buffer. This resides in the RAM as a ROM image and gives a buffer of around 14000 characters instead of the normal 60 or so provided by the Beeb.

Watford's larger module is a different affair. This one comprises two normal sized CMOS RAM chips mounted on a small circuit board about 3in x 2in with two flying leads. One of these is the 'write' lead and connects to a chip on the Beeb's board; the other connects to link S21.

The software that accompanies this sideways RAM module is little different from that with Watford's other model and so there is on the face of it little to distinguish the two models other than the disproportionate size of the larger module. However, the large circuit board allows several options to be fitted, including a rechargeable battery (£3.45) that will keep the contents of your RAM alive even when the Beeb is switched off.

IFEL

IFEL's sideways RAM module is identical in construction to Watford's smaller module. However, the software and manual are different. The same facilities are offered by the IFEL software along with a very useful program to transfer Basic programs to sideways ROM or RAM format. In addition IFEL's manual is a pleasure to use. This is a massive 20 page A5 booklet that goes into memory addressing, the sideways ROM system, and the ROM filing system as well as describe the software itself.

TERRELL

The Terrell sideways RAM module is very similar to the Promicro model. It too is made from two 8K CMOS RAMs mounted piggy-back and packaged up with black tape. The software with this module comes on disc or cassette, in versions for the standard Beeb or one equipped with a ROM board, and provides a familiar set of facilities. ROM images can be saved and loded to disc and those present in your machine listed.

CYGNET

The Cygnet Ramwise sideways RAM module is also based around two piggy-back 8K CMOS RAM chips. This time the whole device is packed in resin inside a tiny plastic case giving a very robust, neat finish.

There are two probe clips to attach to chips on the Beeb's main board with this module. The software comes on disc and provides facilities for loading and saving ROM images, but rather strangely the software cannot distinguish between an empty socket and a sideways RAM module when displaying the ROMs in your machine.

CONCLUSIONS

There isn't much to choose between these modules. They all offer very similar features. Those made up from the half sized CMOS RAM chips are neater and more 'state of the art' but the advantages are very intangible and the disadvantage is a higher price (though IFEL keeps this down to a reasonable level).

All the software does the basic tasks of saving and loading ROM images and listing the ROMs present in your machine at the time. The option of using the RAM as a printer buffer will appeal to many users and this singles out Watford's two boards and IFEL's model. IFEL's also has the useful conversion program for creating ROM images from Basic programs.

If you are only interested in cost then Terrell's model is the cheapest and perfectly serviceable. However, the model with the most features at a reasonable price, and therefore the pick of the bunch, has to be IFEL's.

	Promicro	Watford Large	Watford Small	IFEL	Terrell	Cygnet Ramwise
Price	£29.50	£34.50	£33.35	£26.85	£22.00	£25.95
Address	6 St.George Road Ilford IG1 3PQ	250 High Street Watford Herts	250 High Street Watford Herts	36 Upland Drive Plymouth PL6 6BD	7b Essex Gardens Hornchurch RM11 3EH	PO Box 27 Bordon GU35 OHH
Phone	-	0923-37774	0923-37774	7752-787058	04024-71426	04203-89220
Software	Listing	Disc	Disc	Disc/Cass.	Disc	Disc
List ROMs	*	*	*	*	*	*
Load ROMs	*	*	*	*	*	*
Save ROMs	*	*	*	*	*	*
Buffer		*	*	*		
RFS				*		

1st course
Using Random Numbers

Random numbers play an important role in many programs, from games to serious applications. Cyril Hall explains how random numbers are generated in Basic, and how to exploit them to advantage in your programs.

Most versions of Basic have some facility to generate so-called random numbers, and these have an important role to play in many programs from games to serious applications.

If we throw a dice ten times we may expect results such as:

2 1 6 2 3 4 4 5 1 6

or similar. Providing the dice is unbiased we do not expect any particular pattern or sequence of results to be present or repeated. In fact we assume that each of the six possible results has an equal probability at every throw. This is what we mean by 'random'.

A computer doesn't work on random happenings, being far too logical, but it can generate a long series of apparently unrelated numbers, starting at a number which is called a 'seed'. Using a built-in routine, the computer generates numbers we find unpredictable and random, but as these numbers are not strictly random they are often termed 'pseudo' random numbers.

```
 10 REM RANDOM DICE
 30 MODE 7:PRINT:PRINT
 40:
 50 FOR I=1 TO 10
 60 FOR J=1 TO 10
 70:
 80 dice=RND(6)
 90 PRINT"  ";dice;
110:
120 NEXT J
130 PRINT:PRINT
140 NEXT I
150:
200 END
```

Suppose we wish to simulate the throwing of a dice, i.e. six equally likely but unpredictable whole numbers. These may be generated in BBC Basic by the function RND(6). The program, RANDOM DICE, illustrates its use.

When the program is entered into the Beeb and run, a hundred results simulating the throwing of a dice will be displayed, and further runs will produce a different set of results each time.

An improvement may be made to the program by adding the following lines. Just list the program and then enter:

```
 20 DIM count(6)
100 count(dice)=count(dice)+1
160 FOR K=1 TO 6
170 PRINT;"  ";K;"-";count(K);
180 NEXT K
190:
```

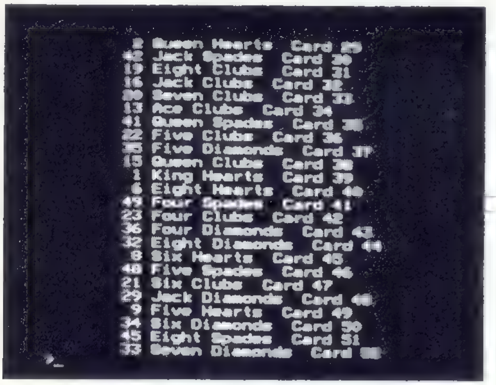
When the extended program is run, it will still generate the hundred results, but it will also count and display how many times each of the six digits was selected so that some assessment of the 'randomness' may be formed.

Suppose we wish to simulate the throwing of a pair of dice. Obviously the results must be between 2 and 12 but they are not equally likely as the probability of scoring seven is 1/6 whilst that of scoring twelve is only 1/36 (because only a pair of sixes will score a 12, but a number of combinations will produce a seven). A good simulation is available by using RND(6)+RND(6) and this may be verified with our RANDOM DICE program by changing three lines as follows:

```
 20 DIM count(12)
 80 dice=RND(6)+RND(6)
160 FOR K=2 TO 12
```

If we wish to produce a result between 2 and 12 with an EQUAL probability of each, then the function RND(11)+1 would do this by selecting a value between one and eleven and adding 1 to each result.

Programs based on card games provide copious opportunities for using random



```
 2 Queen Hearts  Card 29
42 Jack Spades  Card 30
19 Eight Clubs  Card 31
16 Jack Clubs  Card 32
20 Seven Clubs  Card 33
13 Ace Clubs  Card 34
41 Queen Spades  Card 35
22 Five Clubs  Card 36
35 Five Diamonds  Card 37
15 Queen Clubs  Card 38
 1 King Hearts  Card 39
 6 Eight Hearts  Card 40
49 Four Spades  Card 41
23 Four Clubs  Card 42
36 Four Diamonds  Card 43
32 Eight Diamonds  Card 44
 8 Six Hearts  Card 45
48 Five Spades  Card 46
21 Six Clubs  Card 47
29 Jack Diamonds  Card 48
 9 Five Hearts  Card 49
34 Six Diamonds  Card 50
45 Eight Spades  Card 51
33 Seven Diamonds  Card 52
```

numbers. To simulate the drawing, or dealing, of a single playing card we assume that each of the 52 cards has an equal chance and so use RND(52). The numbers that can be generated are allocated one to each card. The program RANDOM CARD will illustrate this.

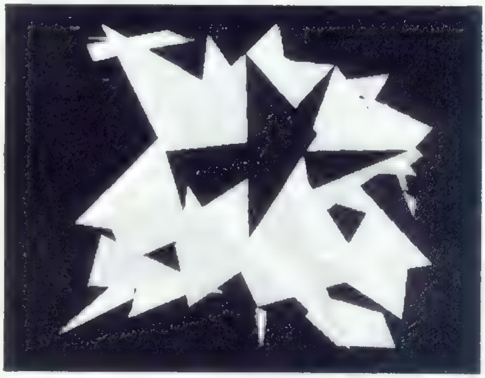
```
   10 REM RANDOM CARD
   20 DIM value$(13),suit$(4)
   30 FOR I=1 TO 13
   40 READ value$(I)
   50 NEXT I
   60 DATA Ace,King,Queen,Jack,Ten,Nine
   70 DATA Eight,Seven,Six,Five,Four
   80 DATA Three,Two
   90 FOR I=1 TO 4
  100 READ suit$(I)
  110 NEXT I
  120 DATA Hearts,Clubs,Diamonds,Spades
  130:
  140 FOR I=1 TO 24
  150 num=RND(52)-1
  160 v=1+num MOD 13:s=1+num DIV 13
  170 PRINT num;" ";value$(v);" ";suit$(s
);TAB(30);"  Card ";I
  180 NEXT I
```

The thirteen possible values and four possible suits are stored in two arrays initially. Each random number in the range 1 to 52 is then split into a value and a suit (line 160) and the appropriate words selected and displayed from the arrays.

This program will deliver the name of a card selected at random twenty four times. However, you may notice that the same card may be chosen more than once. As this would not normally occur in real life, it is desirable to find a way to prevent any card being reselected. The following lines inserted into the program will do this, and to prove the point line 140 is altered so that a full pack of cards is dealt.

```
   20 DIM value$(13),suit$(4),check$(52)
  140 FOR I=1 TO 52
  145 REPEAT
  155 UNTIL check$(num)<>"*"
  175 check$(num)="*"
```

The method used is to have a further 'check' array. Every time a new card is selected this array is checked to see if the same card has already appeared. If it has, another random number is generated and the process repeated. When a new card is accepted, an asterisk is placed in the check array as a marker. You can see the effect of this as the display noticeably slows down as the last few cards are dealt. At this stage the proportion of previously selected cards will be very high. You may find it quite a challenge to see if you can find a faster method.



The random function will speedily generate numbers within a predetermined range so it is a useful source of data for some applications, where choosing and entering many items could be tedious. A simple example in graphics is the production of random triangles or other shapes as in the following program:

```
   10 REM RANDOM TRIANGLES
   20 MODE 1
   30 REPEAT
   40 GCOL 0,RND(3)
   50 PLOT85,RND(1279),RND(1023)
   60 UNTIL FALSE
   70 END
```

The next program, HEADS OR TAILS, uses RND(2) to simulate the spinning of a coin. The output could just have been printed as H T T H H etc., but an effort has been made to improve the presentation. Line 110 keeps a tally of the results which should be approximately equal if sufficient attempts are made.

```
   10 REM HEADS OR TAILS
   20 MODE 7:VDU23,1,0;0;0;0;
   30 CLS:heads=0:tails=0
   40 FOR y=0 TO 1
   50 PRINTTAB(11,y+2)CHR$141;CHR$134;
      "Heads or Tails ?"
   60 NEXT y
   70 PRINTTAB(9,22)CHR$134"Press any key
       to spin"
   80:
   90 REPEAT
  100 num=RND(2)
  110 IF num=1 THEN heads=heads+1
       ELSE tails=tails+1
  120:
  130 PRINTTAB(18,12)SPC5
  140 FOR y=6 TO 10 :PRINTTAB(20,y);
      CHR$124:NEXT
  150 FOR y=6 TO 10 :PRINTTAB(20,y);
      CHR$32:NEXT
  160 IF num=1 THEN PRINTTAB(18,12);
  "HEADS" ELSE PRINTTAB(18,12) "TAILS"
  170:
  180 PRINTTAB(6,19)CHR$131;"Heads = ";
      heads;TAB(25,19)"Tails = ";tails
  190:
  200 G=GET
  210 UNTIL FALSE
  220 END
```

SUMMARISING THE RND FUNCTION

The RND function in BBC Basic has several different formats. The syntax of the function is usually:

variable = RND(number)

RND must be in upper case and no spaces are allowed. num=RND(x) is most useful, providing random whole numbers in the range 1 to x inclusive, whilst num=RND(x)+y produces a similar range of numbers between 1+y and x+y. For example, to generate random whole numbers in the range 100 to 199 use num=RND(100)+99.

RND(1) generates random numbers between 0 and 0.999999. Incidentally, this form was the only version of RND in some earlier versions of Basic.

RND(0) repeats the last number delivered by RND(1) which may be useful when debugging a program with data provided by RND(1).

RND, without argument, generates whole numbers in the range -2147483648 to +2147483647. This is an enormous range and will usually require some form of adjustment to make it useful.

RND(-x) returns the value of -x and resets the generator to a number based on x. This allows the same set of random numbers to be initialised and repeated every time a program is run. This repeatability can be very useful when debugging programs using the RND function. Once the program is working, the RND(-x) statement can be deleted.

At the beginning of the article we mentioned the word 'seed', and the fact that random numbers are generated from a starting number called a seed. Now this can sometimes cause a problem as every time the Beeb is switched on the random number generator is referred to the same seed. If a program using random numbers is run immediately on start up, the generator will repeat exactly the same series of numbers as on each previous occasion.

For some purposes it might not matter that a long series of random numbers had been used previously. Fortunately, the Beeb has an internal clocking device which counts the hundredths of a second which pass from the computer being switched on. The number is held in a variable called TIME, and may be used as follows:

```
   10 num=RND(-TIME)
```

Since it is almost impossible to start running a program at precisely the same moment since switch-on every time, this effectively initialises random number generation with a random number (provided your program has not first reset TIME). Such a line should be included before the normal generation of random numbers.

Random numbers are easy to use and can form the basis of many interesting programs, from implementations of card games (simple and complex) up to more serious applications such as aircraft simulators where random numbers can add that extra touch of realism and unpredictability. Why not give random numbers a chance in your own programs?

by Mitch

ADVENTURE GAMES ADVENTURE GAMES

Village of Lost Souls from Magus, 4, Toronto Close, Durrington, Worthing, West Sussex. Price £9.95 on cassette.

I have spent the last few evenings tramping through a fourteenth century village. This once happy place, is now deserted with only dead bodies and a pack of thieving dogs to bear witness to the devil worshipping rites which have brought disaster to the inhabitants. From the crypt beneath the church to the top of the village windmill, over 140 objects and clues have been strewn to confound any seeker of the truth.

This large text game has a bewildering 200 locations which form a most confusing mapping exercise. To add to your problems a black crow periodically appears to snatch objects from your hands and fly off to its secret hiding place. It's an unusually strong bird as it managed to fly off with my plough! (Reminds me of another very large bird I met in a bar in Portsmouth - but that's another story).

Each location contains the offputting phrase 'Some exits lead ...' forcing you to try exits in every direction from each location in case there is a hidden route. This problem is compounded with the need to EXAMINE everything. Additional objects are quite often not revealed unless the location is completely searched. Taken together these problems serve to make the game quite a test of perseverance. However, some of the problems have multiple solutions so all is not lost should you blunder past the first.

The game has a large vocabulary, but it has many glaring omissions. It is annoying to be told in detail about a room which is full of specific items (e.g. barrels, bell-ropes, planks etc.) to find the game then does not recognise these objects should you attempt to manipulate or examine them. Having found a saw which the game assured me was great for cutting wood, and then found a room full of wood, I was bemused to find the game would not allow me to cut the wood with the saw!

However, otherwise the game is a quality product and full of detail which will require an adventurer, with a large dose of determination, to crack it.

Adventure Description Language from Sigma Press, 5, Alton Rd, Wilmslow, Cheshire SK9 5DY. Book price £8.95.

Writing adventures is great fun, but hard work. Perhaps what you need is a language which is specifically designed to help create your own adventureland. There are many different programming languages to be found in the world of professional computing all of which are designed to be particularly appropriate to a specific field. FORTRAN is favoured by mathematicians, whereas COBOL is the preferred language of data processing. For those would-be wizards amongst you who also have an interest in understanding how and why a new language is created this book will be of special interest.

The ADL language aims to create a very simple set of instructions which can be used to write quite complex games. Like Basic, your program will need an interpreter to execute your commands and control the game. The program listing of the Interpreter is given in the book for you to type in and store on disc. There are also listings for a Screen Editor and Text Compressor to help you write and 'squeeze' the many lines of text you will need to describe the locations and objects in your fantasy world. These programs are available on disc.

A language specifically designed for adventure writing is already used by some of the larger software houses (Infocom and Level 9). To help explain the use of ADL, two games are fully listed and explained showing all the steps in the development. To use the final system you must have a disc drive to hold the interpreter while the game is played. If you want to skip the theory behind the working of the Interpreter and concentrate on writing games you will find it moderately easy to use. However, the book's main appeal will be to students of computer science and those of you who have a slightly more serious interest in computing.

POSTBAG

POSTBAG

Much Ado about Print

When running your second example of the EVAL function (BEEBUG Vol.4 No.8) the numbers printed out in a single column and shot off the top of the screen. So I added ,; at the end of line 23Ø to get results printed to the default field boundaries, 4 columns across the screen. In fact, 26 numbers print neatly and then suddenly field boundaries change and the next 25 are offset (by 6 character positions I think). Every 25 this occurs again. Is this a quirk of Basic, I wonder, or something to do with my system?

Dennis Kemp

In editing programs for the magazine this is something we have seen on several occasions. The default field widths are 1Ø characters wide, so taking 25 numbers, plus the offset of 6 Mr Kemp mentions, gives a total of 256. As most Beeb owners will recognise, this is a significant number in many ways in computing. In this instance it would appear that output is treated as a continuous string, and Basic limits this to 256 characters. The only solution seems to be to force a Return before that total is reached. For example, execute PRINT after every 25 numbers.

Improving the Editor

We have modified the Basic Editor published in BEEBUG Vol.4 No.7 so that there is no need to SPOOL and EXEC to add on the procedure PROCED. All that is needed in the program (or in a menu program) is:

```
   1Ø *FX18
   2Ø *KEY Ø OSCLI("LOAD ED
IT4  "+STR$~(TOP-2))|MEND|M
O.|MM O.7|MPROCED|M
```

The program to be edited is run once. Pressing fØ will append PROCED and the editor can be used as intended (except that Break is no longer used to enter edit mode). On pressing Escape, PROCED is deleted and the program can be run as usual, the only penalty being lines 1Ø and 2Ø above. However, even this is unnecessary if a menu program can execute them instead (for example the Extended Disc Catalogue in BEEBUG Vol.4 No.5). The following lines in the editor also need changing or adding, and the revised version saved as EDIT4:

```
   32ØØ5*KEYØ
   32ØØ6*KEY1
   32Ø2Ø?&8F=MD:*KEY1ØO.|MD
EL.32ØØØ,3246Ø|M
   32225*FX15,1
   32375CLS
   32376*KEY Ø OSCLI("LOAD
ED IT4 "+STR$~(TOP-2))|MEN
D|M O.|MM O.7|MPROCED|M
   32377*KEY1DEL.32ØØØ,324
6Ø|M *KEY1|M
```

Richard & Michael Taylor

Getting a Better View

Perhaps you could tell me why View 2.1 'loses' the printer highlight settings on subsequent SHEETS as in this letter. Also, are BEEBUG, or to your knowledge Acornsoft, going to provide an upgrade from View 2.1 to View 3.Ø?

David T. Crofts

It is a weakness of View 2.1 that it does not fully carry highlight information across sheet boundaries. Acorn say that this has been cured in View 3.Ø (see our recent series on View for more information - BEEBUG Vol.4 Nos.9 & 1Ø). Both View 1.4 and View 2.1 can be upgraded to View 3.Ø by sending the old chip together with a cheque for £23 to Valarie Raworth, Acornsoft, Cambridge Technopark, 645 Newmarket Road, Cambridge CB5 8PD.

In return you will receive View 3.Ø, a Function Key Card and a Reference Card. The new View 3.Ø manual will cost a further £11.5Ø. No new Printer Driver Generator is needed. This upgrade is only available from Acornsoft, NOT from BEEBUG.

More than a Trace

The Improved Trace facility published in BEEBUG Vol.3 No.7 is disabled on a soft Break. Tape users, for whom re-loading the program is a nuisance, can re-enable the facility by entering:

```
?&DØØ=&AD <Return>
CALL &DØØ <Return>
```

D.A.Stevens

HINTS HINTS HINTS HINTS HINTS HINTS HINTS

ARIES TURN-ON (OR OFF)

The following !BOOT file will turn on or off the Aries B-20 board before loading in a program ('MENU' in this case). Change XON to XOFF as needed.

```
    *K.10 CHAIN "MENU"
    *K.0 *XON|MCALL&D9CD|M
    *FX15,0
    *FX138,0,128
```

David Davies

SIDEWAYS RECOVERY

Many sideways ROM boards (such as ATPL's) have the facility for sideways RAM to occupy the highest priority socket. If partially completed or un-debugged ROM software is occupying this position when Break is pressed, the OS may enter the software via its language entry and the machine hang. No amount of pressing of Break or Ctrl-Break will cure this as the software is in the highest priority socket. The solution is to run the following program before using the sideways RAM. This places the machine code equivalent of *BASIC onto disc as a !BOOT file so that Shift-Break will restore your Beeb to Basic.

```
    10 P%=&2000:[ LDA #187:
LDX #0:LDY #255 JSR &FFF4:L
DA #142:JSR &FFF4:]
    20 *SAVE !BOOT 2000 201
0 2000
    30 *OPT 4,2
```

Philip Endecott

MERGING VIEWSTORE FILES

Viewstore files cannot be simply merged together to form a single file. The solution to this problem is to READ the separate files into VIEW and then SAVE the combined file again. This can be IMPORTed back into Viewstore format noting that 13 is the record separator and 9 the field separator.

E.A. Spencer

PASSING PARAMETERS TO WORDWISE PLUS

Although the manual describes how variables may be passed to machine code routines from Wordwise Plus no mention is given of how to pass values back again. The answer is to poke the values directly into memory where Wordwise Plus stores them. This is at the same location as Basic, i.e. A% to Z% from &404 onwards, each variable using four bytes.

Brian Storey

PRINTER CHECK

When a program requires the use of a printer, the following line at the beginning of the program will check that the printer (serial or parallel) is switched on, on line, and ready to receive data, and so prevent the machine hanging up at the crucial moment.

```
    10 VDU2:PRINT':VDU3:A=I
NKEY(10):IF ADVAL(-4)<63 PR
INT "PRINTER NOT ON LINE"
```

Stephen Wilcock

INITIALIZING A DATA AREA

To initialize a data area in a machine code program to contain (say) zero the following function may be used:

```
    1000 DEFFNdata(N%,opt)
    1010 FOR R%=1 TO N%
    1020 ?P%=0
    1030 P%=P%+1
    1040 NEXT
    1050 =opt
```

The function can be called directly from your program in the position required with parameter N% being the length of the data area and opt the current value for OPT using a statement such as:

```
  OPT FNdata(&100,pass)
```

Stephen Mellor

SINGLE KEY UNDERLINE

The following function key definition enables single words to be easily underlined in Wordwise Plus.

```
    *KEY0 |!|\|!!US|!"|!|}|
!,|!!UE|!"
```

Press f0 directly after the word to be underlined. This automatically inserts the necessary embedded codes before and after the word.

W.A. Simcock

DATE CHECKING

The following (fairly) simple function will check the validity of a date where D,M,Y hold the day, month and year. A zero is returned for an invalid date. The function will not cope with the non-leap year in 2100 but hopefully a better routine will be published in BEEBUG nearer the time.

```
DEFFNvalidate(D,M,Y):=D*M*
(M<13)*(D<31+((M+(M>7))AND
1)+(M=2)*(2+((Y AND 3)=0)))
```

David Abbot

Across the Tube (Part 2)

Ian Trackman concludes his detailed description of the Tube and its inner workings with a particular look at the use of interrupts.

The Tube software communicates across the Tube using an eight byte block of memory on each side of the Tube. These bytes are arranged as four pairs of 8-bit registers. Any value placed in a register on one side of the Tube is automatically transferred to the corresponding register on the other side, and thereby made accessible to the other processor.

Last month we looked at several official and semi-official methods of transferring data from the control of the Second Processor to that of the Host Processor. Now I shall look at communication initiated by the Host Processor in the opposite direction. However, as the Tube is essentially symmetrical and communication across it is a two-way process, it should be noted that communication from Second Processor to Host Processor is also possible by methods similar to those described here.

THE TUBE REGISTERS

Each pair of registers comprises a status register and a data transfer register, though only the top two bits of the status registers are used. The program in the Host processor waits for bit 6 of the status register to be set (using a loop conditional on the BVC command). Once bit 6 is set, the Host Processor places the data to be transferred in the data register, and sets bit 7 of the status register to signify that it has done so and that the data register is ready to be read by the Second Processor.

Meanwhile, the Second Processor performs a similar process. It sets bit 6 of the status register to signify that it is ready to receive some data, and waits for bit 7 to be set (with a BPL loop) before attempting to read that data. The whole process is very similar to the handshaking procedure used to communicate with a printer across the printer port.

The four pairs of registers have specific purposes allotted to them:

1. OSWRCH calls and Host Processor events.
2. OSBYTE, OSWORD, OSFILE, OSBPUT, etc.
3. Data transfer between the Host and Second Processors.
4. Control of data transfer.

One complication is that the four pairs of registers do not have the same addresses on both sides of the Tube. In the Host Processor the eight bytes are located as follows:

OSWRCH status register	&FEE0
OSWRCH data register	&FEE1
OSBYTE, etc. status register	&FEE2
OSBYTE, etc. data register	&FEE3
Data transfer status register	&FEE4
Data transfer data register	&FEE5
Control status register	&FEE6
Control data register	&FEE7

In the Second Processor the corresponding bytes are located between &FEF8 and &FEFF.

The first two pairs of registers are reserved for OS use of the Tube. We have already looked at using these for data transfers from Second Processor to Host Processor. Instead, I shall now concentrate on the use of the third pair of registers for transferring data from Host to Second Processor.

USING THE REGISTERS

It is possible to use the registers directly, but only if you are certain that no other parts of the system are using the Tube. However, it is the function of the Tube software to use these registers correctly, so it is only sensible to make use of that software unless you have an application that demands the maximum speed data transfer across the Tube.

Incidentally, accessing the two data transfer registers generates an interrupt in the Second Processor. This is passed to the Tube software in the Second Processor which handles the data transfer before returning control to the main program in the Second Processor.

The Tube software resides between &400

and &6FF in the Host Processor. It has one main entry point at &406 to handle data transfers, and that is our main interest.

CLAIMING AND RELEASING THE TUBE

As a first step, you might want your software to ensure that the Tube is present and active. An OSBYTE &EA call with X=Ø and Y=&FF will return with X=&FF if the Tube is operative and with X=Ø if it is not.

Your software should now 'claim' the Tube to ensure that no other caller has control over it (e.g. a filing system call in operation, or an interrupt requiring Tube service). To claim the Tube, call &406 with the accumulator set to &CØ+ID. The identification number (ID) is a 6-bit value (i.e. in the range Ø to &3F). The OS and standard filing systems use numbers from zero as follows:

Ø	-	Cassette filing system
1	-	Disc filing system
2	-	Econet low level primitives
3	-	Econet filing system
4	-	ADFS
5	-	Teletext

I suggest that you use an ID from the other end of the range, e.g. &3F. A call to &406 returns with the carry set if the claim has been accepted. So, to claim the Tube with an ID of &3F, the code would be:

```
    .claim
    LDA #(&CØ+&3F)
    JSR &4Ø6
    BCC claim
```

After use, the Tube should be released with another call to &406 and with the accumulator set to &80+ID.

```
    .release
    LDA #(&8Ø+&3F)
    JSR &4Ø6
```

When claiming and releasing the Tube the values of the X and Y registers are immaterial.

READING AND WRITING DATA

Besides ensuring that there is no conflict between Tube users, the Tube software also offers a number of ready-made data transfer options. These are accessed by specifying a number in the accumulator when calling the Tube software at &406.

Ø	-	read one byte
1	-	write one byte
2	-	read two bytes
3	-	write two bytes
4	-	JMP to Second Processor address in response to OSCLI or *FX142
5	-	not used
6	-	read &1ØØ bytes
7	-	write &1ØØ bytes

When making such calls to the Tube software, the X and Y registers are used to point to a four-byte indirection block in the Host Processor which in turn contains the 32 bit address in the Second Processor where data is to be written to or read from. The top 16 bits of this address should be set to Ø for the standard 6502 Second Processor.

The three read and write options (for 1, 2, and &1ØØ bytes) operate in similar ways. For speed considerations, it is sensible to use the option most suited to the number of bytes to be transferred.

You have to allow time for the Tube software to handle the transfers. There must be a pause between each successive read/write operation, dependent on the option used:

1 byte	24 micro-seconds
2 byte	13 micro-seconds per byte
&1ØØ bytes	1Ø micro-seconds per byte

An additional pause must also be included in the case of transfers from the Second Processor to the Host Processor immediately after calling the Tube Software at &4Ø6. These additional pauses should be:

1 byte	24 micro-seconds total
2 byte	26 micro-seconds total
&1ØØ bytes	19 micro-seconds total

There are a number of ways of generating these pauses. One method is to loop with a decrementing counter; another to use several JSR-RTS sequences. The latter method has the advantage of leaving all the registers and flags unaffected. The JSR-RTS sequence takes 12 clock cycles, so that doing this once at 2MHz will generate a 6 micro-second delay. Here is a sample routine to read &1ØØ bytes from the Second Processor. It assumes that the 32 bit address in the Second Processor has been set up at 'block' and that 'destination' is a two byte indirection location in zero page of the Host Processor pointing to the start of memory where the data is to be stored.

```
    .claim LDA #(&C0+ID)      \set up my ID
    JSR &406               \claim the tube
    BCC claim        \again if unsuccessful
    LDX #block MOD &100       \point to the
    LDY #block DIV &100     \address block
    LDA #6              \&100-byte transfer
    JSR &406
    JSR ret                          \6 us
    JSR ret                         \12 us
    JSR ret                         \18 us
    LDY #0  \initialize index (now 19 us)
    .loop LDA &FEE5      \read byte (2 us)
    STA (destination),Y   \store it (3 us)
    NOP                        \pause 1 us
    NOP                        \pause 1 us
    NOP                        \pause 1 us
    INY                   \next byte (1 us)
    BNE loop  \1.5 us - 10.5 us now passed
    LDA #(&80+ID)
    JSR &406                  \release Tube
    .ret RTS
```

Note: the letters 'us' above represent micro-second.

JUMPING TO THE SECOND PROCESSOR

By calling the Tube software with the accumulator set to 4, the Second Processor will jump to the address in the Second Processor specified in the address block in the Host Processor pointed to by the X and Y registers when making the call. However, unlike the data transfer calls, where the user is responsible for releasing the Tube, this call automatically releases the Tube after execution.

Typically, this call is used by the filing system to execute a *RUN request. The call does not return control on the Host processor side but goes back to running the normal Host Processor Tube software.

HANDLING INTERRUPTS ACROSS THE TUBE

Second Processor interrupts are for internal data transfer and event-signalling purposes only. We shall now see how to handle interrupts which would normally be accessible to the programmer in the Host Processor.

The first step to handling Host Processor interrupts is to set up an interrupt-handling routine in the normal way, installing the code in the Host Processor as described last month. Since this is likely to be a small program dedicated to handling the interrupt, the 'clean' approach is to split the process in two. The interrupt-handler itself sets, say, bit 7 of a flag byte and immediately hands back control to the OS. The remainder of the program, after initialising the flag bit, loops around waiting for the flag bit to be set. Don't forget to include an over-riding exit condition in the loop - such as the Escape key being pressed - to prevent it running on when the foreground program has stopped.

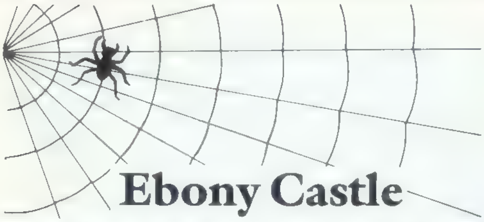
When an interrupt is detected, the interrupt-handler must notify the Second Processor that the interrupt has occurred. Two possible methods suggest themselves. Firstly, a byte could be written to a specified address in the Second Processor. This byte could be regularly polled by the foreground control program in the Second Processor. When its value changes, the Second Processor program could divert to a separate routine.

The second method is to set up an event-handling routine in the Second Processor and then generate an event with OSEVEN (&FFBF) when the interrupt is detected. The event should preferably be a 'user event' with the Y register set to 9 before making the call.

If you use the first method, your program can then use other OS calls to obtain more information, if needed, about the status of relevant parts of the system. This information could also be gathered by the interrupt-handler in the Host Processor, and then be transferred across to storage locations in the Second Processor. When the second method is used, such OS calls would have to be issued from within the event handling routine. This is not to be recommended.

Also not recommended, but possibly even faster, a third method of interrupting the Second Processor would be to use the fact that the Tube transfer itself generates an interrupt in the Second Processor. This would have to be re-vectored into your own routine, taking extreme care to handle properly any subsequent interrupt which might occur before your own interrupt handling had finished. It would also, of course, be necessary to cater for any 'official' Tube traffic.

→49



Ebony Castle

Jonathan Temple is the author of several highly professional games published in recent issues of BEEBUG. This latest challenge could be his best yet.

You, a simple serf, have been elected leader of a secret band plotting to overthrow the evil Necromancer. The Necromancer, hearing of your plans, has imprisoned you in the dungeons of the infamous Ebony Castle, from which he is sure you will not be able to escape...

However, you have managed to get out of the dungeons and are now hiding in the main kitchens of the ruined castle. You must now try and escape to the countryside to re-join your comrades so that the Necromancer can finally be banished from the realm.

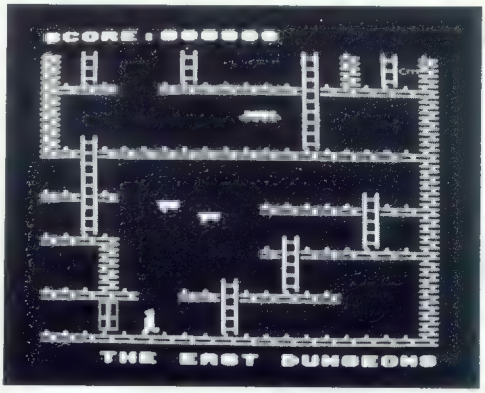
The castle is inhabited by huge spiders and, from the very first second as a spider comes racing towards you, you can tell your task is not going to be easy. On occasions you'll need split-second timing to beat them.

There are difficult jumps to contend with as well, and it's so cold in the castle that you'll also have to be wary of treacherous icy platforms.

To make your task even more difficult, several areas of the castle and the countryside are shut off by coloured doors. To open these you'll have to obtain the corresponding coloured keys which have been cunningly scattered throughout the castle.

The program features the famous 'interlocking screens' of games like Jet Set Willy, where you can walk between rooms at will - as long as you're carrying the correct keys!

Because the game is already challenging there is no set time limit as such, but if you do actually manage to reach the countryside the bonus awarded depends on how long you took as well as how many lives you have remaining.



As with many games, you start with three lives. The keys you'll need to use are the usual 'Z', 'X', '*' and '?' for left, right, up and down movement respectively, whilst pressing 'Return' will let you jump. In addition the game can be paused by pressing 'P', and continued with 'C'.

When the game is paused, 'S' and 'Q' will turn the sound on and off respectively, whilst 'R' will allow you to re-start - useful if you become trapped, although you do lose a life.

THE PROGRAM

The program should be fairly straightforward to enter, although the data for the five rooms (lines 2700-2890) should be entered carefully, along with the data for the keys, doors and castle layout (lines 2280-2360). If not entered correctly, some of these lines could quite easily make the whole game impossible! Also, since the program runs in mode 2, disc users (or anyone else with PAGE over &E00) will need to include these few lines at the start of the program to move it down.

```
1 IF PA.<&E01 THEN 10
2 *K.0 P.'"Moving down..."|L|M*T.|M
  F.A%=0TO(TOP-PA.)S.4:A%!&E00=A%!PA.
  :N.|MPA.=&E00|MO.|MDEL.1,4|MRUN|M
3 *FX 138,0,128
4 END
```

PROCEDURES USED

init	Initializes keys, doors, and the castle layout
chars	Defines characters
envs	Defines four sound envelopes
title	Displays title page
setvars	Sets variables, changes flashing colours to white and resets keys
screen	Draws current screen and sets certain variables
read	Reads in data for PROCscreen
man	Checks for keys pressed, moves man
jump	Makes man jump
new	Looks after man moving between screens
key	Looks after collecting keys
doors	Prints doors if locked
pause	Looks after pause feature
spider	Moves the lift and checks for collision
score	Displays player's score and keys collected
killed	Looks after death of your man and resets variables
end	Prints "Game Over" message, pauses
print	Prints text in 3-D
outside	Draws countryside and plays tune
feature	Draws cloud or tree foliage

For those interested in writing their own platform games, or who wish to design their own screens for the game (or make some of the screens easier!) the data for the five rooms is held in a heavily compacted format in lines 2700-2890.

First comes a string, the name of the room, which is printed beneath the platforms during play. The first number given is the number of platforms, whilst the second, third and fourth affect the colours to be used in the screen - the colour of the bricks, mortar and ladders respectively.

The next six numbers relate to the spider, giving the x-position, y-position, limit x-position, limit y-position, horizontal direction and vertical direction of the spider respectively.

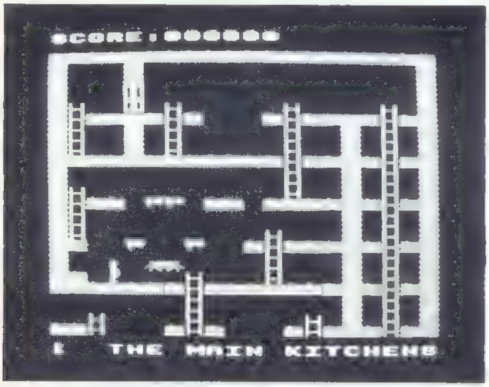
These are followed by the data for the platforms themselves, in the form of a long string. Each platform is held in a string of four ASCII characters, with the first two letters representing the x-1 and y+2 tab positions of the platform (@=0, A=1, B=2 etc.). So A@ would mean at 0,2; AA would mean at 0,3; BB would mean at 1,4; and so on.

Following the two letters is a hexadecimal number. This gives the length, direction and type of each platform, by separating the information into discrete bits. Bits 0-4 hold the length (0-31), bit 5 holds the direction (0=horizontal, 1=vertical) and bits 6 and 7 hold the type where:-

0 is a brick platform
1 is an ice platform
2 is a ladder

This may all seem very complicated, but it does mean that varied and complex screens can be stored in only a few lines of data, and this cuts down drastically on typing in.

```
   10 REM PROGRAM EBONY CASTLE
   20 REM VERSION B0.1
   30 REM AUTHOR  J. Temple
   40 REM BEEBUG  MAY 1986
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 ON ERROR GOTO 3720
  110 MODE 2
  120 PROCtitle:PROCinit
  130 PROCchars:PROCenvs
  140 REPEAT:PROCsetvars
  150 REPEAT:PROCscreen(P%)
  160 REPEAT:PROCman
  170 PROCspider:UNTIL E%
  180 IF E%=1 PROCkilled
  190 UNTIL Z%=0 OR E%=2
  200 IF E%=2 PROCoutside
  210 PROCend:UNTIL FALSE
  220 END
  230 :
 1000 DEFPROCman
 1010 A%=X%:B%=Y%:C%=V%:D%=W%
 1020 IFINKEY-56 PROCpause
 1030 G%=POINT(X%+8,Y%-68):H%=POINT(X%+4
8,Y%-68)
 1040 IFINKEY-74 IFJ%+F%=0 IFG%+H%>0 J%=
6:U%=(INKEY-98)-(INKEY-67):SOUND 18,2,10
,5
 1050 IFJ% PROCjump:ENDPROC
 1060 IFG%+H%=0 Y%=Y%-32:W%=W% EOR 1:F%=
F%+1:GOTO1130
 1070 IFF% SOUND 17,1,1,1:SOUND 18,0,0,0
:IFF%>3 E%=1 ELSE IFF% F%=FALSE
```



```
 1080 IFG%=8 OR H%=8 IFINKEY-98+INKEY-67
=0 X%=X%+(V%=233)*32-(V%=232)*32:W%=W% E
OR1:GOTO1130
 1090 IFINKEY-98 IFPOINT(X%-8,Y%)<1 X%=X
%-32:Y%=Y%-(POINT(X%+24,Y%-32)>0)*32:W%=
W% EOR 1:SOUND 18,-10,50,1:IFV%<>233 V%=
233:W%=236
 1100 IFINKEY-67 IFPOINT(X%+64,Y%)<1 X%=
X%+32:Y%=Y%-(POINT(X%+36,Y%-40)>0)*32:W%
=W% EOR 1:SOUND 18,-10,50,1:IFV%<>232 V%
=232:W%=234
 1110 IFINKEY-73 IFPOINT(X%+8,Y%)=LC% Y%
=Y%+32:V%=V% EOR 1:W%=W% EOR 1:IFC%<>238
 V%=238:W%=240
 1120 IFINKEY-105 IFPOINT(X%+8,Y%-68)=LC
% Y%=Y%-32:V%=V% EOR 1:W%=W% EOR 1:IFC%<
>238 V%=238:W%=240
 1130 IFD%<>W% GCOL3,15:MOVE A%,B%:VDUC%
,10,8,D%:MOVE X%,Y%:VDUV%,10,8,W%
 1140 IFPOINT(X%,Y%-20) PROCkey
 1150 IFX%>1216 OR X%<0 OR Y%>928 OR Y%<
188 PROCnew
 1160 ENDPROC
 1170 :
 1180 DEFPROCjump
 1190 X%=X%+A%(J%)*U%:Y%=Y%+B%(J%)
 1200 J%=J%-1:GCOL 3,15:MOVE A%,B%
 1210 VDUC%,10,8,D%:MOVE X%,Y%
 1220 VDUV%,10,8,W%
 1230 IFY%MOD32=28 IFPOINT(X%,Y%-68)+POI
NT(X%+56,Y%-68)>0 J%=0
 1240 IFPOINT(X%,Y%-20) IFP%<5 PROCkey
 1250 IFU%=1 IFPOINT(X%+64,Y%-32)>0 U%=0
 1260 IFU%=-1 IFPOINT(X%-8,Y%-32)>0 U%=0
 1270 IFJ%=0 IFPOINT(X%+8,Y%-68)<1 IFPOI
NT(X%+48,Y%-68)<1 F%=1
 1280 ENDPROC
 1290 :
 1300 DEFPROCnew
 1310 L%=-(X%>1216)-(X%<0)*2-(Y%>928)*3-
(Y%<188)*4
 1320 P%=R%(P%-1,L%-1):U%=-U%
 1330 IFL%=1 X%=0 ELSE IFL%=2 X%=1216
 1340 IFL%=3 Y%=188 ELSE IFL%=4 Y%=924
 1350 IFP%<6 PROCscreen(P%) ELSE E%=2
 1360 ENDPROC
 1370 :
 1380 DEFPROCkey
 1390 IFX%<>K%(P%-1,1) ORY%<>K%(P%-1,2)
ENDPROC
 1400 SOUND 17,4,100,1:K%(P%-1,3)=1
 1410 MOVE X%,Y%:GCOL3,K%(P%-1,0):VDU242
 1420 PROCscore(5000)
 1430 RESTORE:FOR L%=1 TO P%
 1440 READ M%:NEXT
 1450 D%(M%-1,3)=0:IFM%=P% PROCdoors
 1460 ENDPROC
 1470 :
 1480 DATA 3,1,4,2,5
 1490 :
 1500 DEFPROCdoors
 1510 GCOL0,0
 1520 IFD%(P%-1,3) GCOL0,D%(P%-1,0)
 1530 MOVE D%(P%-1,1),D%(P%-1,2)
 1540 VDU 227,10,8,228,10,8,229
 1550 ENDPROC
 1560 :
 1570 DEFPROCspider
 1580 IFABS((R%+16)-Y%)<48 IFABS((Q%+32)
-X%)<96 E%=1
 1590 GCOL3,2:MOVE Q%,R%:VDU230,231
 1600 Q%=Q%+SX%:R%=R%+SY%
 1610 MOVE Q%,R%:VDU230,231
 1620 IFSX% IFQ%=S1% OR Q%=S3% SX%=-SX%
 1630 IFSY% IFR%=S2% OR R%=S4% SY%=-SY%
 1640 IFABS((R%+16)-Y%)<48 IFABS((Q%+32)
-X%)<96 E%=1
 1650 ENDPROC
 1660 :
 1670 DEFPROCpause
 1680 I=TIME:REPEAT N%=GET AND &DF
 1690 IF N%=81 THEN *FX 210,1
 1700 IF N%=83 THEN *FX 210,0
 1710 UNTIL N%=67 OR N%=82
 1720 TIME=I:IF N%=82 E%=1
 1730 ENDPROC
 1740 :
 1750 DEFPROCscore(N%)
 1760 S%=S%+N%:VDU4,17,7,31,6,0
 1770 PRINT LEFT$("000000",6-LEN(STR$(S%
)))+STR$(S%);"  ";
 1780 FOR L%=0 TO4
 1790 VDU 17,K%(L%,0):IFK%(L%,3) VDU242
 1800 NEXT
 1810 VDU5
 1820 ENDPROC
 1830 :
 1840 DEFPROCkilled
 1850 Z%=Z%-1:SOUND 0,1,100,2
 1860 FOR I=1 TO 2000:NEXT
 1870 X%=192:Y%=316:V%=232:W%=234:P%=3
 1880 ENDPROC
 1890 :
 1900 DEFPROCend
 1910 *FX 15,Ø
 1920 VDU4,28,4,15,15,11,12,26,5
 1930 PROCprint("GAME OVER",352,636,1,3)
 1940 PROCprint("<SPACE>",416,572,4,6)
 1950 REPEAT UNTIL GET=32
 1960 ENDPROC
 1970 :
 1980 DEFPROCprint(T$,X,Y,A,B)
 1990 GCOL Ø,A:MOVE X,Y:PRINT T$
 2000 GCOL Ø,B:MOVE X-8,Y-4:PRINT T$
 2010 ENDPROC
 2020 :
 2030 DEFPROCsetvars
 2040 Z%=3:S%=Ø:P%=3:TIME=Ø
 2050 X%=192:Y%=316:V%=232:W%=234
 2060 FOR L%=9 TO 15:VDU 19,L%,7;Ø;
 2070 NEXT
 2080 FOR L%=Ø TO 4
 2090 K%(L%,3)=FALSE:D%(L%,3)=1
 2100 NEXT
 2110 ENDPROC
 2120 :
 2130 DEFPROCinit
 2140 DIM W$(2),A%(6),B%(6),K%(4,3),D%(4
,3),R%(4,3)
 2150 C$=CHR$17
 2160 W$(1)=C$+CHR$8+C$+CHR$128+CHR$225
 2170 RESTORE2280
 2180 FOR L%=1 TO 6:READ A%(L%),B%(L%)
 2190 NEXT
 2200 FOR L%=Ø TO 4:FOR M%=Ø TO 2
 2210 READ K%(L%,M%):NEXT,
 2220 FOR L%=Ø TO 4:FOR M%=Ø TO 2
 2230 READ D%(L%,M%):NEXT,
 2240 FOR L%=Ø TO 4:FOR M%=Ø TO 3
 2250 READ R%(L%,M%):NEXT,
 2260 ENDPROC
 2270 :
 2280 DATA Ø,-32,32,-32,32,Ø,32,Ø,32,32,
Ø,32
 2290 DATA 3,768,892,5,256,188
 2300 DATA 4,192,700,2,512,188
 2310 DATA 1,1152,924
 2320 DATA 5,Ø,220,2,1088,220
 2330 DATA 3,256,860,4,384,348
 2340 DATA 1,192,220
 2350 DATA 2,6,Ø,Ø,3,1,Ø,4,Ø,2,Ø,5
 2360 DATA 5,Ø,2,Ø,Ø,4,3,Ø
 2370 :
 2380 DEFPROCscreen(P%)
 2390 VDU4,12,19,1,1;Ø;19,5,5;Ø;19,8,6;Ø
;23;10,32;Ø;Ø;Ø;17,7
 2400 PRINTTAB(Ø,Ø)"SCORE:"
 2410 PROCscore(Ø):VDU4,17,7
 2420 RESTORE 2700:IFP%>1 PROCread
 2430 READ T$,N%,A%,B%,LC%
 2440 READ S1%,S2%,S3%,S4%,SX%,SY%,W$
 2450 W$(Ø)=C$+CHR$A%+C$+CHR$(128+B%)+CH
R$224
 2460 W$(2)=C$+CHR$LC%+C$+CHR$128+CHR$22
6
 2470 PRINTTAB(Ø,30);STRING$(Z%-1,CHR$23
2);TAB(3,30) T$;
 2480 FOR L%=1 TO N%*4-3 STEP 4
 2490 A$=MID$(W$,L%,4)
 2500 N1%=EVAL("&"+MID$(A$,3))
 2510 N2%=N1% AND31:N3%=N1% AND32
 2520 N4%=(N1% AND192)DIV64
 2530 D$="":IFN3% D$=CHR$(1Ø)+CHR$(8)
 2540 PRINTTAB(ASC(A$)-65,ASC(MID$(A$,2)
)-62) STRING$(N2%-1,W$(N4%)+D$)+W$(N4%)
 2550 NEXT
 2560 E%=Ø:F%=Ø:J%=Ø:VDU5:PROCdoors
 2570 IFK%(P%-1,3)=Ø MOVE K%(P%-1,1),K%(
P%-1,2):GCOL3,K%(P%-1,Ø):VDU242
 2580 GCOL3,15:MOVE X%,Y%:VDUV%,1Ø,8,W%
 2590 Q%=S1%:R%=S2%:GCOL3,2
 2600 MOVE Q%,R%:VDU230,231
 2610 ENDPROC
 2620 :
 2630 DEFPROCread
 2640 FOR L%=1 TO P%-1
 2650 READ T$,N%,A%,B%,C%
 2660 READ S1%,S2%,S3%,S4%,SX%,SY%,W$
 2670 NEXT
 2680 ENDPROC
 2690 :
 2700 DATA "THE ENTRANCE HALL"
 2710 DATA 29,2,4,5
 2720 DATA 1Ø88,196,192,196,-32,Ø
 2730 DATA A@14AA36TA2ABDØ3MDØ1PDØ4FEØ1H
F29JG41LHØ1NI41BJØ6PJØ4RNØ3BOØCTO2BPQØ3S
TØ1GUØ7BVØ3RVØ1PW41AZ14CCA7QCA7EIA6QMA4C
NACLNA7
 2740 DATA "  BANQUETING HALL"
 2750 DATA 31,5,4,3
 2760 DATA 32Ø,316,1Ø88,316,32,Ø
 2770 DATA A@14AA2ATA35JDØABEØ3GEØ1JHØ4P
IØ4BJØ3GK41AN2DBNØ6KNØ1NO41QPØ3IRØ3BSØ4N
SØ1ET22BV13GW23NW23BZ13CDA6RCA6KGA7CMA6R
OA7JQAACUA6PYA2
 2780 DATA "THE MAIN KITCHENS"
 2790 DATA 31,1,3,6
 2800 DATA 832,322,128,322,-32,Ø
 2810 DATA A@14AA36EA22TA3ACFØ6LFØ8EG23P
G33BJ12CNØ2FN42INØ2LNØ8BRØ1ER41HR41KRØ9B
UØ1BV12AZØ3GZØ3MZØ7BEA5GEA5MEA9REB6BMA5L
QA5HUA6CYA2NYA2
 2820 DATA "THE WEST DUNGEONS"
 2830 DATA 2Ø,6,4,3
 2840 DATA 64,188,1Ø24,188,32,Ø
 2850 DATA A@3BG@33T@2ABDØANDØ6BI12LJ28B
M13SQØ2BRØBRT22GVØ3LV42PVØ5BWØ3FWØ2BZ13C
@B7J@B2P@A4
 2860 DATA "THE EAST DUNGEONS"
 2870 DATA 24,4,2,6
```

```
2880 DATA 640,812,640,700,0,-16
2890 DATA A@2AP@23T@3BBC03GC0DBI13AM04G
N41LN08I041AQ04DR24LR08AV05HV08AZ14C@A3H
@A3N@A9R@A3CHA9QMA5MQA5JUA5
2900 :
2910 DEFPROCoutside
2920 PROCscore(Z%*20000):IFTIME<30000 P
ROCscore((30000-TIME)*10)
2930 VDU4,28,0,26,19,2,17,132,12,26
2940 VDU28,0,28,19,26,17,130,12,26
2950 VDU17,7,17,128
2960 PRINTTAB(3,30)"THE COUNTRYSIDE !";
2970 FOR N%=150 TO 450 STEP 150
2980 VDU24,N%-20;125;N%+20;350;18,0,129
,16,26
2990 PROCfeature(N%,400,90+RND(20),90+R
ND(40),13+RND(3),2,3)
3000 NEXT
3010 PROCfeature(700,800,150,50,9,7,4)
3020 PROCfeature(830,750,180,50,12,7,4)
3030 VDU5,18,3,15:X%=1216:Y%=252
3040 MOVE X%,Y%:VDU V%,10,8,W%
3050 FOR X%=1216 TO 592 STEP -16
3060 MOVE X%,Y%:VDU V%,10,8,W%
3070 W%=W% EOR 1
3080 MOVE X%-16,Y%:VDU V%,10,8,W%
3090 FOR I=1 TO 150:NEXT
3100 SOUND 18,-10,50,1
3110 NEXT:VDU4
3120 PRINTTAB(3,30)"CONGRATULATIONS !";
3130 RESTORE 3210:N%=81:*FX 15,0
3140 FOR L%=1 TO 10:READ A%,D%:N%=N%+A%
3150 SOUND 1,-10,N%,D%
3160 SOUND 2,-5,N%+48,D%:NEXT
3170 REPEAT UNTIL ADVAL(-7)=15
3180 FOR I=1 TO 2000:NEXT
3190 ENDPROC
3200 :
3210 DATA 0,4,8,4,8,4,4,4,8,8,-12,8,4,8
,-12,8,8,8,-16,8
3220 :
3230 DEFPROCfeature(X%,Y%,SX%,SY%,L%,C1
%,C2%)
3240 VDU29,X%;Y%;:MOVE0,0:S2%=0
3250 MOVE SX%+SX%/L%,0:S1%=SX%+SX%/10
3260 FOR I=0 TO 6.3 STEP .1
3270 X%=SX%*COS(I)+SX%/L%*COS(I*L%)
3280 Y%=SY%*SIN(I)+SY%/L%*SIN(I*L%)
3290 GCOL 0,C1%:MOVE32,12:PLOT85,X%,Y%
3300 MOVE S1%,S2%:GCOL0,C2%:DRAW X%,Y%
3310 S1%=X%:S2%=Y%:NEXT:VDU29,0;0;
3320 ENDPROC
3330 :
3340 DEFPROCtitle
3350 VDU5
3360 PROCprint("EBONY CASTLE",256,896,1
,3)
3370 PROCprint("Can YOU meet",224,736,4
,6)
3380 PROCprint("the Challenge?",160,672
,4,6)
3390 PROCprint("<SPACE>",384,352,4,5)
3400 REPEAT UNTIL GET=32
3410 ENDPROC
3420 :
3430 DEFPROCchars
3440 VDU23,224,223,223,223,0,-5,-5,-5,0
3450 VDU23,225,-1,-1,127,110,98,32,32,0
3460 VDU23,226,66,66,66,66,126,66,66,66
3470 VDU23,227,0,-1,-1,153,153,153,153,
153
3480 VDU23,228,153,153,153,153,-1,-5,-1
,153
3490 VDU23,229,153,153,153,153,153,-1,-
1,0
3500 VDU23,230,0,55,11,63,78,149,36,34
3510 VDU23,231,0,236,208,-4,114,169,36,
68
3520 VDU23,232,48,56,48,48,32,48,40,40
3530 VDU23,233,12,28,12,12,4,12,20,20
3540 VDU23,234,40,56,40,48,32,32,32,48
3550 VDU23,235,40,56,40,48,40,168,200,1
2
3560 VDU23,236,20,28,20,12,4,4,4,12
3570 VDU23,237,20,28,20,12,20,21,19,48
3580 VDU23,238,24,26,26,126,88,88,24,28
3590 VDU23,239,24,88,88,126,26,26,24,56
3600 VDU23,240,20,20,20,22,16,16,16,48
3610 VDU23,241,40,40,40,104,8,8,8,12
3620 VDU23,242,0,0,96,159,149,149,96,0
3630 ENDPROC
3640 :
3650 DEFPROCenvs
3660 ENVELOPE 1,1,0,0,0,0,0,0,90,-1,-2,
-3,97,97
3670 ENVELOPE 2,133,8,4,8,3,1,1,126,0,0
,-10,80,0
3680 ENVELOPE 3,3,-1,-1,-1,70,50,1,100,
-1,-1,-10,101,5
3690 ENVELOPE 4,133,8,4,8,3,1,1,126,0,0
,-10,126,0
3700 ENDPROC
3710 :
3720 MODE7:REPORT:PRINT" at line ";ERL
3730 *FX15,1
3740 END
```

←44 There is no doubt that communicating across the Tube is not an easy affair. However, I hope that these notes will enable you to experiment for yourself and to make the most of this complex and little-known part of the BBC micro.

ACKNOWLEDGEMENTS

I am grateful to Hugo Tyson, Roger Wilson, Jon Thackray, and David Bell at Acorn Computers for their valuable assistance in the preparation of this article.

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.

Editor: Mike Williams
Assistant Editor: Geoff Bains
Production Assistant:
Yolanda Turuelo
Technical Assistant: Alan Webster
Secretary: Debbie Sinfield
Managing Editor: Lee Calcraft
Additional thanks are due to Sheridan Williams, Adrian Calcraft, John Yale and Tim Powys-Lybbe.

Editorial Address
BEEBUG
PO BOX 50,
Holywell Hill,
St. Albans AL1 3YS

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £40 per page, but please give us warning of anything substantial that you intend to write. A leaflet, 'Notes of Guidance for Contributors' is available on receipt of an A5 (or larger) SAE.

In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "View", or other means, but please ensure an adequate written description of your contribution is also included. If you use cassette, please include a backup copy at 300 baud.

HINTS

There are prizes of £5 and £10 for the best hints each month, plus one of £15 for a hint or tip deemed to be exceptionally good.

Please send all editorial material to the editorial address above. If you require a reply it is essential to quote your membership number and enclose an SAE.

SUBSCRIPTIONS

Send all applications for membership, subscription renewals, subscription queries and orders for back issues to the subscriptions address.

MEMBERSHIP SUBSCRIPTION RATES

£ 6.90 6 months (5 issues) UK ONLY)
£12.90 UK - 1 year (10 issues)
£19.00 Europe, £22.00 Middle East
£24.00 Americas & Africa, £26.00 Elsewhere

BACK ISSUES
(Members only)

Vol	Single issues	Volume sets (10 issues)
1	90p	£8
2	£1	£9
3	£1.20	£11
4	£1.20	–

Please add the cost of post and packing as shown:

DESTINATION	First issue	Each subsequent issue
UK	30p	10p
Europe	70p	20p
Elsewhere	£1.50	50p

All overseas items are sent airmail (please send a sterling cheque). We will accept official UK orders but please note that there will be a £1 handling charge for orders under £10 that require an invoice. Note that there is no VAT on magazines.

Back issues are for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

Subscriptions, Back Issues &
Software Address

BEEBUG
PO BOX 109
St. Johns Road
High Wycombe HP10 8NP

Hotline for queries and software orders

St Albans (0727) 40303
Manned Mon-Fri 9am-4.30pm

24hr Answerphone Service for Access and Barclaycard orders, and subscriptions
Penn (049481) 6666

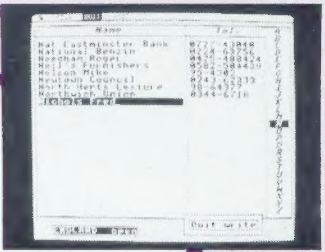
If you require members' discount on software it is essential to quote your membership number and claim the discount when ordering.

Magazine Cassette/Disc

MAY 1986 CASSETTE DISC CONTENTS

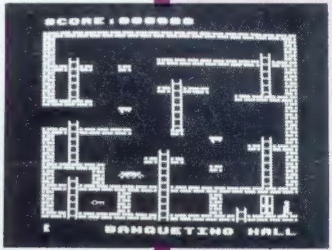
MANDELBROT GRAPHICS — a fascinating and colourful exploration.
WINDOWS AND ICONS — complete suite of programs for the Phone Book including all the USI utilities from parts 1 and 2.
MAIL-MERGE FOR FILER — complete BEEBUG Filer program including latest update.
SOFT SCREEN SHUFFLE — adding that slick professional touch to your programs.
DO-IT-YOURSELF BASIC — complete program with all this month's additions to Basic.
*SPOOL GRAPHICS — routine to turn all your Basic graphics into machine code.
BEEBUG WORKSHOP — extended demonstration of stacks and queues.
FIRST COURSE — seven examples of the use of random numbers.
EBONY CASTLE — the latest quality game from games author Jonathan Temple. This could be his best yet.

EXTRA FEATURES THIS MONTH
MAGSCAN — data for this issue of BEEBUG (Vol. 5. No. 1).

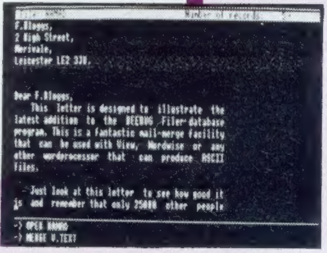


```
Name                          Tel.
Nat Eastminster Bank          0727-43048
National Benzin               0224-63756
Needham Roger                 0425-488424
Neil's Furnishers             0582-584430
Nelson Mike                   75-4302
Newtown Council               0243-62313
North Herts Leisure           98-64377
Northwich Union               0344-6718
```

The Phone Book



Ebony Castle



```
F.Bloggs,
2 High Street,
Merivale,
Leicester LE2 3JB,


Dear F.Bloggs,
    This letter is designed to illustrate the
latest addition to the BEEBUG Filer-database
program. This is a fantastic mail-merge facility
that can be used with View, Wordwise or any
other wordprocessor that can produce ASCII
files.

    Just look at this letter to see how good it
is and remember that only 25000 other people
-) OPEN NAMES
-) MERGE V.TEXT
```

Mail-Merge with Filer

All this for £3.00 (cass) £4.75 (disc) + 50p p&p
Back issues (disc since Vol. 3 No. 1, cass since Vol. 1 No. 10) available at the same prices.

Subscription rates	DISC UK	CASS UK	DISC O'seas	CASS O'seas
6 months (5 issues)	£25.50	£17	£30	£20
12 months (10 issues)	£50	£33	£56	£39

Prices are inclusive of VAT and postage as applicable.
Sterling only please.

Cassette subscriptions can be commuted to disc subscription on receipt of £1.70 per issue of the subscription left to run.

All subscription and individual orders to
BEEBUG, PO BOX 109, St. Johns Road, High Wycombe HP10 8NP

Why Pay £2.50 for a FREE Demo Disc?

What You Get With Your £2.50 Free Disc

1. A stunning demonstration of many programs from the Beebugsoft range of software. Hear for yourself how good Studio 8 is. Watch Icon Master and Toolkit Plus at work. See the results of Hershey characters and RomIt. And much more.

2. If you decide to purchase any of our programs once you have seen the demonstrations, send us the disc label to receive a discount of £3.00 off the members discounted price of any one item of software from the latest 20 page full colour Beebugsoft catalogue. Hence the disc is free!

3. Also on the disc is a free arcade style machine code game. Blast the monsters with "Grid Runner".

4. A special code-breaker program is included on the disc. Issues of Beebug and Acorn User, up to July 1986 will include special code numbers; type these numbers into your code-breaker program to see if you are one of the lucky winners for that month. Each winner will receive the Beebugsoft program of their choice.

5. Once you have finished using our Free Disc, you have a top quality disc which you may re-format and use for your own purposes.

6. SAVE A FURTHER £1.00 Combine your order for the Demo Disc with an order for Beebugsoft software and you may immediately deduct £1.00 from the order.

You Can't Go Wrong! Just fill in your name and address below and send it to us with £2.50.

This month's code-breaker number is V11698

This offer is limited to one disc per household/institution

Please send me a "free" disc,
I enclose a cheque for £2.50/Please debit my Barclaycard/Access No. ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐
OR I also enclose an order for software, please charge only £1.50

Name ______________________________

Address ______________________________

____

Specify 40 or 80 Track ______________

Send to:
Beebugsoft,
PO Box 109,
High Wycombe,
Bucks., HP10 8NP